JRun Setup Guide

JRun 3.0 for Windows®, UNIX, &
Linux™

Allaire Corporation

Copyright Notice

Part number: AA-JRSET-RK

Contents

PREFACE

Before You Begin

This chapter provides an overview of the JRun installation procedure and describes the hardware and software requirements for installing JRun. This chapter also provides guidelines on accessing JRun and Allaire resources such as Web sites, documentation, and technical support.

Contents

JRun Product Variations

JRun is a Java application server supporting the latest servlet/JSP and EJB specifications from Sun Microsystems. JRun ships in the following versions:

- Developer. Free for non-commercial use for developing and testing Web applications and EJBs. Not licensed for deployment. JRun Developer allows an unlimited number of Java Virtual Machines (JVMs), 3 concurrent connections for servlets and JSPs, and 3 concurrent connections for EJBs.
- Professional. Priced per processor for commercial deployment. JRun Professional allows an unlimited number of JVMs and concurrent connections for servlets and JSPs.
- Enterprise. Priced per processor for developing and deploying Enterprise-class applications. JRun Enterprise includes HTTP-based load balancing and failover software using Allaire ClusterCATS. It allows an unlimited number of JVMs and concurrent connections for servlets, JSPs, and EJBs.
- Studio. Priced per license. An integrated JSP development environment based on the HomeSite HTML editor. Does not include the JRun server.

Contact Allaire for the latest information on pricing.

System Requirements for Installing JRun

This section lists the hardware and software requirements for installing JRun.

Hardware requirements

A full installation of JRun has the following minimum hardware requirements:

- 32 MB RAM (64 recommended)
- 20 MB hard disk space (50 recommended)

Software requirements

JRun has the following software requirements (which are detailed below):

- System running either Windows or UNIX
- Netscape Communicator or Internet Explorer
- Java Runtime Environment (JRE) 1.1 (version 1.2.2 or later required for EJB, JTA and JMS)

Operating system requirements

JRun requires the following minimum operating system versions. Note that some JVMs and Web servers have more stringent OS requirements.

- Windows 95/98/NT/2000 (SP 3 or greater with NT)
- Solaris 2.6, 2.7, 8
- Red Hat Linux 6.x
- HP-UX 11.0
- IBM AIX 4.2, 4.3
- SGI IRIX 6.5
- Compaq UNIX Tru64 4.0

Internet browser requirements

JRun includes the JRun Management Console (JMC), an HTML utility for configuring the JRun environment and the connection between JRun and your Web server. Since the JMC is Web-based, you need to be running one of the following Web browsers:

- Netscape Communicator Version 4.0 or later
- Internet Explorer Version 4.0 or later

Java requirements

The following table lists the Java utilities that you need to develop Java servlets, JSP pages, and EJBs. The minimal requirements are listed in the table. However, we recommend you use the latest release available, and that you do not use Beta releases of Java utilities for a production system.

You can obtain these utilities at the following URL:

```
http://java.sun.com/products/jdk/1.2/
```

JRun Java Software Requirements

Java Component	Windows	UNIX
Java Runtime Environment (JRE)	JRE included with JRun; however, you can use your own as well. Must be 1.1.6 or later.	Must obtain a JRE (included with the Java JDK). Must be 1.1.6 or later.
Java Virtual Machine (JVM)	JVM Version 1.2 is included with JRun; however, you can use Version 1.1.8 or later.	Must obtain JVM Version 1.1.8 or later for most flavors of UNIX. Version 1.2 is preferred (and required for HP/UX).
Java servlets	Any Java JDK with Java compiler.	Any Java JDK with Java compiler.

JRun Java Software Requirements (Continued)

Java Component	Windows	UNIX
JavaServer Pages (JSP)	No additional software required; JRun includes all required tools.	No additional software required; JRun includes all required tools.
Enterprise JavaBeans (EJB)	JDK 1.2 or later.	JDK 1.2 or later.

The following table lists the utilities included with JRun for each supported platform:

JRun Supplied Utilities

Utility	Description	Windows	UNIX
Rhino	An open-source JavaScript interpreter and compiler.	1.4	1.4
jikes	An open-source JavaServer Pages compiler.	1.06	1.06

Supported JVMs

The following table lists the JVMs supported by JRun. Contact Allaire for the latest updates:

Supported JVMs

JVM	JDK	System	Vendor URL
Sun Microsystems 1.1.8	1.1.8	Windows NT/ x86	`http://java.sun.com/`
Sun Microsystems 1.2.1 & 1.2.2	1.2.2	Windows NT/ x86	`http://java.sun.com/`
Sun Microsystems 1.3	1.3	Windows NT/ x86	`http://java.sun.com/`
Sun Java HotSpot 1.0.1	1.2.2	Windows NT/ x86	`http://java.sun.com/`
Microsoft VM 2.02	1.1	Windows NT/ x86	`http://www.microsoft.com/`

Supported JVMs			
JVM	**JDK**	**System**	**Vendor URL**
Microsoft VM 4.0	1.1.4	Windows NT/ x86	`http://www.microsoft.com/`
IBM JDK 1.1.8	1.1.8	Windows NT/ x86	`http://www.ibm.com/`
IBM 1.1.8	1.1.8	Linux/i686	`http://www.ibm.com/`
Blackdown 1.1.7Bv3	1.1.7B	Linux/x86	`http:// java.blackdown.org/ java-linux.html`
Blackdown 1.2.2 rc3, 4	1.2.2	Linux/i386	`http://java.sun.com/`
Sun/Borland 1.2.2 RC2	1.2.2 RC2	Linux/i386	`http://java.sun.com/`
Sun 1.1.6	1.1.6	Solaris/Sparc	`http://www.sun.com`
Sun 1.2.1	1.2.1	SunOS/Sparc	`http://www.sun.com`
Sun 1.2.1_0	1.2.1	SunOS/Sparc	`http://www.sun.com`
Sun 1.2.1_04 pre-release	1.2.1	SunOS/Sparc	`http://www.sun.com`

Upgrading From a Previous Version of JRun

Upgrading to JRun 3.0 allows you to take advantage of the many additions to JRun including EJB support, an updated servlet API, and an enhanced GUI administration tool. Some of these additions require preparation to take full advantage of them.

Note If you are installing JRun 3.0 over a previous Beta release of the product, you should uninstall your current version before continuing with the installation.

This section introduces you to the issues you might encounter when upgrading from a previous version of JRun to 3.0.

For more information about the differences from previous implementations of the JSP, EJB, and servlet specifications, refer to Sun's specifications.

Running 2.3.x and 3.0 at the same time

By default, the JRun installation script installs JRun into `C:\Program Files\Allaire\JRun` (Windows) and `/opt/jrun` (UNIX). If you are upgrading from a previous version of JRun, you must first stop all JRun servers. Then, you should do one of the following:

- Move your existing JRun directory to a new location
- Install your new version of JRun into a new location

If you do not uninstall your existing copy of JRun, stop all JRun and Java processes prior to installing JRun on Windows 95/98/NT. Otherwise, JRun may incorrectly configure the JRun Web Server.

Note Allaire does not recommend installing JRun 3.0 and 2.3.x on the same machine.

To stop JRun processes on Windows NT:

1. Click Control + ALT + Delete. The NT Security window appears.
2. Click Task Manager. The Windows NT Task Manager Appears.
3. Select the Processes tab.
4. Sort by Image Name.
5. Stop the following processes:

```
javaw.exe
jrun.exe
```

Administration

Interface

Instead of the Swing-based administration utility used in JRun 2.3.x, you now use the JRun Management Console (JMC), a browser-based utility for configuring JRun. For information on using the JMC, refer to Chapter 3.

Property files

The JMC stores values for initialization and configuration of JRun in property files. The number of property files has been reduced from more than 75 to less than 10 for the default installation. For more information on JRun property files, refer to Chapter 4.

Web applications

With the introduction of the version 2.2 Servlet specification comes the concept of Web applications and `.war` files. The `.class` and supporting files within Web applications are deployed into a directory hierarchy dictated by this specification. JRun still supports using individual servlets that are not part of Web applications by having a `<`*`jrun_rootdirectory`*`>/servlet` directory. The `default` Web application includes this directory in its classpath.

Deprecated and removed functions

JRun 3.0 implements the latest specifications, from Web applications to Enterprise JavaBeans. However, some functionality is being dropped or phased out. This section describes some of these deprecated features.

CF_Anywhere

JRun continues to enable you to process files that use a subset of Allaire's ColdFusion markup language (CFML). However, this functionality will not be supported in future releases of JRun. For more information, refer to the `JRun Developer Center`.

Server Side Includes (SSI)

While SSIs were once a common method of creating dynamic content, this functionality in JRun is included primarily to support older implementations. JavaServer Pages (JSP) and Java servlet technology have replaced and greatly expanded upon the capabilities of SSI.

Active Server Pages

JRun no longer supports Active Server Pages (ASP).

Installation Overview

This section describes the basic procedure for installing and configuring JRun. Note that this procedure varies based on your Web server, Web server version, and Web server platform.

The basic procedure for installing and configuring JRun is shown in the following table:

Installation Steps

Step	Chapter
1. Install JRun.	Chapter 1
2. Confirm that JRun is installed properly.	Chapter 1
3. Configure your Web server for communicating with JRun.	Chapter 2
4. Confirm that your Web server and JRun are communicating.	Chapter 2
5. Perform any additional JRun configuration using the JRun Management Console (JMC).	Chapter 3

Java Products Overview

This section provides an overview of the most recent versions of the major Java products. For more information, refer to Sun’s Web site at `http://java.sun.com`.

Java platform

The Java Platform defines the architecture of the Java environment. There are three editions of the Java 2 Platform:

- Java 2 Platform, Standard Edition (J2SE)
- Java 2 Platform, Enterprise Edition (J2EE)
- Java 2 Platform, Micro Edition (J2ME)

The Java 2 Platforms are implemented by the Java Software Development Kits, described below.

Java Software Development Kit

The Java Software Developer Kit (SDK) is also commonly referred to as the Java Development Kit (JDK). It consists of the Java Runtime Environment (JRE) plus tools and core classes for developers to compile, debug, and run applications for the Java platform. On Windows systems, the JRE is included with the SDK. On UNIX, the JRE is not included in the same downloaded file. The SDK cannot be distributed per the licensing agreement.

Major components

- Compiler and debugger
- Java Runtime Environment (JRE)
- Win32 Performance Pack (optional)
- Solaris Native Threads Pack (optional)

Versions

- JDK 1.0.*x*
- JDK 1.1
- J2 SDK 1.2.2 Standard Edition (also known as the Java 2 SDK)
- J2 SDK 1.2.2 Enterprise Edition (also known as the Java 2 SDK)

The J2 SDK *Enterprise Edition* adds support to the SDK for advanced services including JSP, EJB, and servlets.

Java Runtime Environment

The Java 2 Runtime Environment (J2 JRE) is an implementation of the Java Virtual Machine (JVM) specification that comes with a set of supporting classes. It contains everything necessary to run programs written for the Java 2 platform. Unlike the SDK, developers can freely distribute the JRE per the licensing agreement.

Major components

- Java Virtual Machine (JVM)
- Java application launcher
- Runtime class libraries
- Java Plug-in (for browsers)
- Symantec JIT Compiler (Windows) or Sun JIT (UNIX)

Versions

- JRE 1.1
- JRE 1.2.2

A JVM is the software implementation of a CPU designed to run compiled Java code. Many companies produce their own JVMs, including HP, Sun, Microsoft and Symantec. The term *Java Runtime Environment* (JRE) is the Sun-specific name for their JVM implementation. It is commonly used to describe JVMs from any vendor. In this document, JRE and JVM are used interchangeably. For a list of JVMs supported by JRun, refer to “Supported JVMs” on page 4.

Advanced services

Java is an extensible language and is continually providing increased functionality. This section describes some of the extensions that JRun supports. During JRun installation, you can choose to install or not install each of these components.

Servlets

Servlets are Java Web components that generate dynamic content. JRun 3.0 conforms to the Servlet Specification 2.2 from Sun. This specification is built on 2.1 and includes support for Web applications and Web application archives (WARs). JRun’s implementation of Servlet 2.2 requires JRE 1.1 or higher.

JavaServer Pages

JavaServer Pages (JSPs) are an extension of the Java Servlet API. They combine Java code with HTML to create dynamic Web pages. JRun 3.0 supports JSP Specification 1.1 from Sun. This spec is based on 1.0 and includes support for tag extensions and containers. JRun’s implementation of JSP 1.1 requires JRE 1.1 or higher.

Enterprise JavaBeans

Enterprise JavaBeans (EJB) is the distributed, server-side, component-based software architecture for the J2EE Platform. JRun supports the Enterprise JavaBeans Specification 1.1 from Sun. The EJB 1.1 spec extends 1.0 with development and deployment enhancements such as JTA and JMS. JRun's implementation of EJB requires JRE 1.2.2 or higher.

Developer Resources

Allaire Corporation is committed to setting the standard for customer support in developer education, documentation, technical support, and professional services. Our Web site is designed to give you quick access to our entire range of online resources. The table below shows the locations of these resources.

Allaire Developer Services	
Resource	**Description**
Allaire Web site `www.allaire.com`	General information about Allaire products and services.
Information on JRun `www.allaire.com/products/jrun/`	Detailed product information on JRun and related topics.
Developer Community `www.allaire.com/developer`	All of the resources you need to stay on the cutting edge of JRun development, including online discussion groups, Component Exchange, Resource Library, technical papers, and more.
JRun Dev Center `www.allaire.com/developer/ jrunreferencedesk/`	A one-stop information resource for servlet resources, development tips, articles, documentation, and white papers.
Technical Support `www.allaire.com/support`	Allaire offers a wide range of professional support programs, including telephone-based support, Web-based support, and the Allaire Knowledge Base, which contains hundreds of technical articles relating to all Allaire products and describing various tips, techniques, and workarounds. In addition, the Installation Support System provides solutions to the most common installation issues for your configuration.

Allaire Developer Services (Continued)

Resource	Description
JRun Support Forum `forums.allaire.com/jrunconf`	Access to experienced JRun developers through participation in the Allaire Online Forums, where you may post messages and read replies on many subjects relating to JRun.
Professional Education `www.allaire.com/education`	Information about classes, on-site training, and online courses offered by Allaire.
Consulting Services `www.allaire.com/consulting`	Allaire Consulting offers services targeted at areas that can most influence the success of a Web application development effort.

About JRun Documentation

The JRun documentation set contains the following:

- *Release Notes*
- *JRun Setup Guide*
- *Developing Applications with JRun*
- *JRun Samples Guide*
- *Using Allaire ClusterCATS* (ships with JRun Enterprise)
- *Using JRun Studio* (ships with JRun Studio)

Online documentation

Allaire provides online versions of all JRun manuals as Adobe Acrobat (PDF) files. The PDF files are included on the JRun CD and installed in the JRun `/` directory by default. You can access them by clicking on the Product Documentation link in the JRun Management Console's Welcome screen.

You can download the Adobe Acrobat files from the Allaire Web site at `www.allaire.com/documents`.

Documentation conventions

When reading these documents, note the following formatting cues:

- Numbered steps indicate procedures.
- Code samples, filenames, and URLs are set in a `monospaced` font.
- Notes and tips are identified by **bold** type in the margin.

- Bulleted lists present options and features.
- Menu levels are separated by the greater than (>) sign.

Other Resources

You may want to consult the following resources for more information on topics covered in this document:

Books

- *Java Servlets* by Karl Moss, published by McGraw Hill, 1999, ISBN: 0071351884
- *Java Servlets: By Example* by Alan R. Williamson, published by Manning Publications, 1998, ISBN: 188477766X
- *Java Servlet Programming* by Jason Hunter and William Crawford, published by O'Reilly & Associates, 1998, ISBN: 156592391X
- *Developing Java Servlets* by James Goodwill, published by Sams, 1999, ISBN: 0672316005
- *Inside Servlets: Server-Side Programming for the Java Platform* by Dustin R. Callaway, published by Addison-Wesley Pub. Co., 1999, ISBN: 0201379635
- *Professional Java Server Programming*, published by Wrox Press Ltd., 1999, ISBN: 1861002777
- *Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition* by Ed Roman, published by Wiley, ISBN: 0471332291
- *Enterprise JavaBeans* by Richard Monson-Haefel, published by O'Reilly & Associates, ISBN: 1565928695

Online resources

- Java servlet API (http://java.sun.com/products/servlet)
- JavaServer Pages (http://java.sun.com/products/jsp)
- Servlet Source (http://www.servletsource.com)
- JSP Resource Index (http://www.jspin.com)
- ServerPages.com (http://www.serverpages.com)
- Enterprise JavaBeans (`http://java.sun.com/products/ejb/`)

Contacting Allaire

Corporate headquarters

Allaire Corporation
One Alewife Center
Cambridge, MA 02140

Telephone: 617.761.2000
Fax: 617.761.2001

`http://www.allaire.com`

Technical support

Telephone support is available Monday through Friday from 8 AM to 8 PM Eastern time (except holidays).

Toll Free: 888.939.2545 (U.S. and Canada)

Telephone: 617.761.2100 (outside U.S. and Canada)

Postings to the JRun Support Forum (`http://forums.allaire.com`) may be made at any time.

Sales

Toll Free: 888.939.2545

Telephone: 617.761.2100

Fax: 617.761.2101

E-mail: sales@allaire.com

Web: `http://www.allaire.com/store`

CHAPTER 1

Installing JRun

This chapter describes how to install JRun. After performing the steps in this chapter, you should configure your Web server for communicating with JRun as described in Chapter 2.

Contents

Pre-Installation Checklist

JRun 3.0 functions as both a stand-alone Java application server and a plug-in module that adds Java application support to an existing Web server.

If you intend to connect JRun to an external Web server, complete the following table to ensure that you have the required information available.

Pre-Installation Checklist

Property	Description	Value
Web server	Record the Web server (for example, NES, Apache, IIS).	
Web server version	Record the Web server's version.	
Proxy host IP address	Record the IP address of the host that JRun uses to connect to your Web server. If JRun and your Web server are hosted on the same machine, use 127.0.0.1. If JRun and the Web server are hosted on different machines, use the IP address of JRun's host.	

Installing JRun

This section describes how to install JRun on the following systems:

- Windows 95/98/NT/2000 installation on page 16
- UNIX and Linux installation on page 25

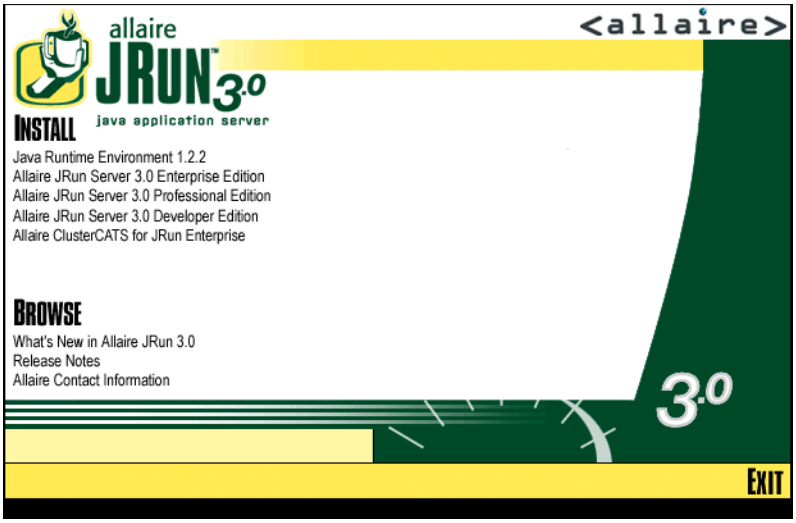
Windows 95/98/NT/2000 installation

This section describes how to install JRun on a Windows 95/98/NT system.

To install JRun:

1. Stop your Web server if you are going to connect JRun to it.
2. Exit all currently running Windows applications.
3. Execute the JRun installation file `jr30w.exe`.

The JRun splash screen appears.



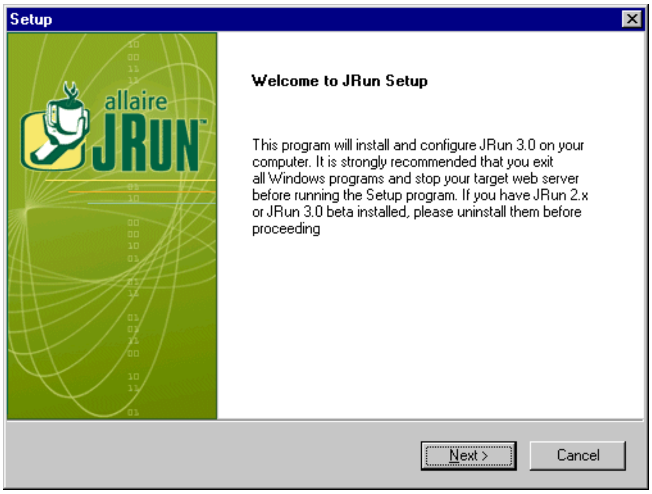
Java Runtime Environment 1.2.2
Allaire JRun Server 3.0 Enterprise Edition
Allaire JRun Server 3.0 Professional Edition
Allaire JRun Server 3.0 Developer Edition
Allaire ClusterCATS for JRun Enterprise

What's New in Allaire JRun 3.0
Release Notes
Allaire Contact Information

4. Click on the version of JRun 3.0 you would like to install under the Install section.

Note You must have JRE 1.2.2 or higher to use the Enterprise JavaBeans (EJB) components. If you do not have JRE 1.2.2 and want to use EJB, you should install it now by clicking Install Java. This will install Sun's J2 JRE version 1.2.2. Then, resume JRun installation.

The Welcome window appears.

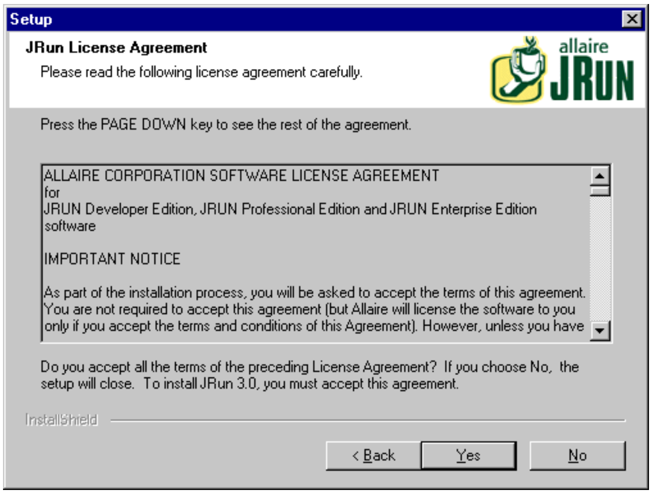


Welcome to JRun Setup

This program will install and configure JRun 3.0 on your computer. It is strongly recommended that you exit all Windows programs and stop your target web server before running the Setup program. If you have JRun 2.x or JRun 3.0 beta installed, please uninstall them before proceeding

5. Click Next.

   The License Agreement window appears.



Setup

JRun License Agreement

Please read the following license agreement carefully.

Press the PAGE DOWN key to see the rest of the agreement.

ALLAIRE CORPORATION SOFTWARE LICENSE AGREEMENT
for
JRUN Developer Edition, JRUN Professional Edition and JRUN Enterprise Edition
software

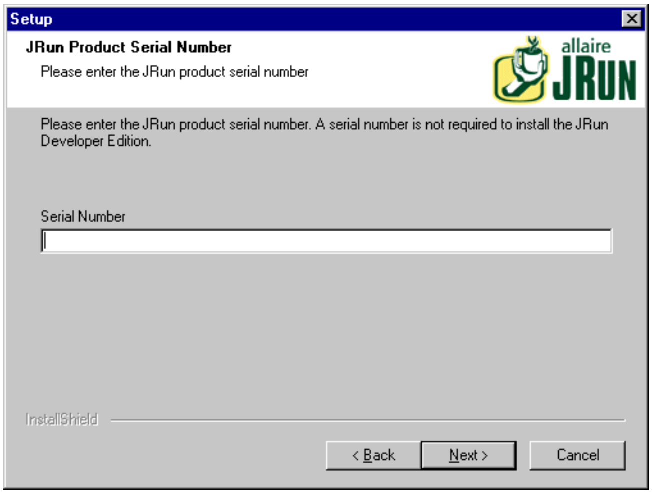
IMPORTANT NOTICE

As part of the installation process, you will be asked to accept the terms of this agreement. You are not required to accept this agreement (but Allaire will license the software to you only if you accept the terms and conditions of this Agreement). However, unless you have

Do you accept all the terms of the preceding License Agreement? If you choose No, the setup will close. To install JRun 3.0, you must accept this agreement.

6. Click Yes to accept the JRun License Agreement or No to cancel the installation.

   The Product Serial Number window appears.



Setup

JRun Product Serial Number

Please enter the JRun product serial number

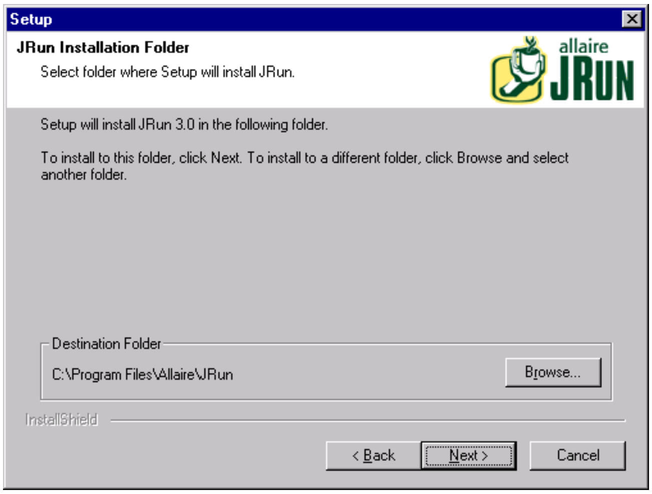
Please enter the JRun product serial number. A serial number is not required to install the JRun Developer Edition.

Serial Number

7. Enter the serial number *exactly* as it was provided to you by Allaire and click Next.

   If you are installing the JRun Developer Edition or an evaluation version of JRun, leave the field blank (the default). If you are upgrading from a previous version of JRun, enter your new serial number. You will then be prompted for your old JRun version 2.*x* license key.

The Installation Folder window appears.



JRun Installation Folder

Select folder where Setup will install JRun.

Setup will install JRun 3.0 in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

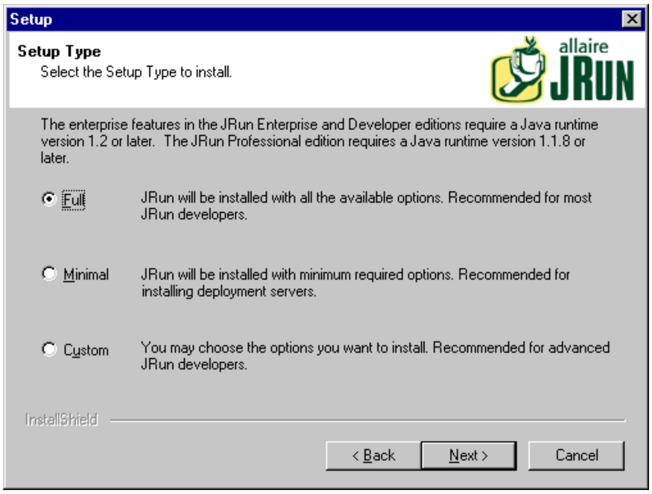
Destination Folder

C:\Program Files\Allaire\JRun

8. Select a destination folder for JRun and click Next.

Note This folder will be referred to as <*jrun_directory*> in this document.

The Setup Type window appears.



Setup Type

Select the Setup Type to install.

The enterprise features in the JRun Enterprise and Developer editions require a Java runtime version 1.2 or later. The JRun Professional edition requires a Java runtime version 1.1.8 or later.

[✓ Full] JRun will be installed with all the available options. Recommended for most JRun developers.

Minimal JRun will be installed with minimum required options. Recommended for installing deployment servers.

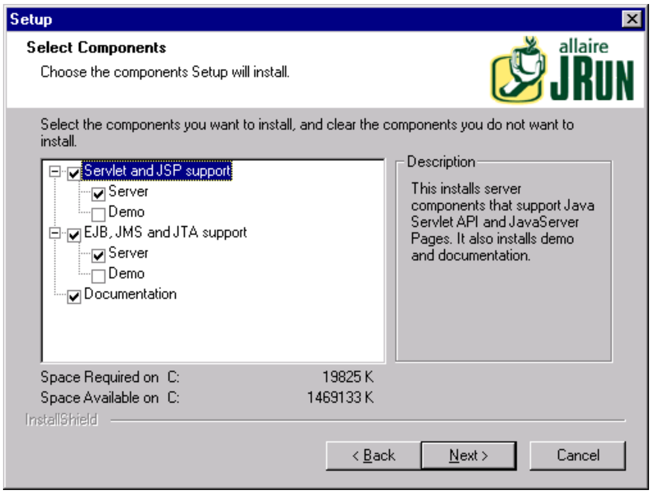
Custom You may choose the options you want to install. Recommended for advanced JRun developers.

9. Select the type of installation and click Next. The following table explains these options.

Custom Component Selection

Option	Description
Full	Installs all the available options. This includes servlet, JSP, EJB, JMS, and JTA support, as well as samples and documentation. This option is recommended for most JRun developers.
Minimal	Installs the minimum required options. This includes servlet, JSP, EJB and JMS support. Documentation and samples are not installed. This option is recommended for installing deployment servers.
Custom	Allows you to choose the options you want to install. This option is recommended for advanced JRun developers.

If you selected Custom, the Select Components window appears. For JRun Professional users, the EJB components are not available.



Select Components

Choose the components Setup will install.

Select the components you want to install, and clear the components you do not want to install.

- [✓] Servlet and JSP support
  - [✓] Server
  - [ ] Demo
- [✓] EJB, JMS and JTA support
  - [✓] Server
  - [ ] Demo
- [✓] Documentation

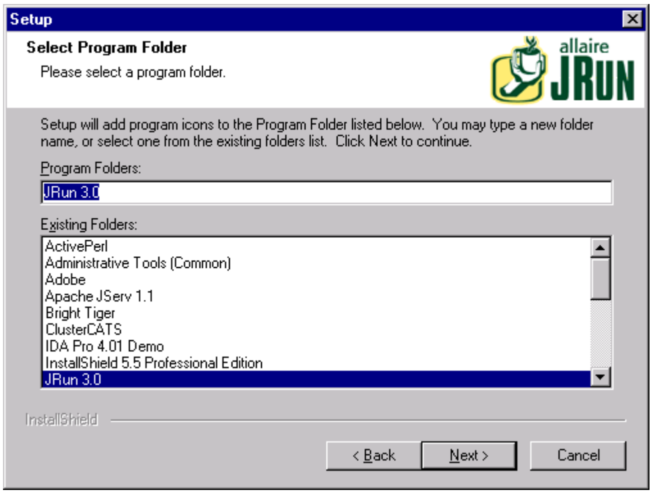
Description

This installs server components that support Java Servlet API and JavaServer Pages. It also installs demo and documentation.

Space Required on C:	19825 K
Space Available on C:	1469133 K

10. Choose the desired JRun components and click Next.

The Select Program Folder window appears.


Setup

Select Program Folder

Please select a program folder.

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.

Program Folders:

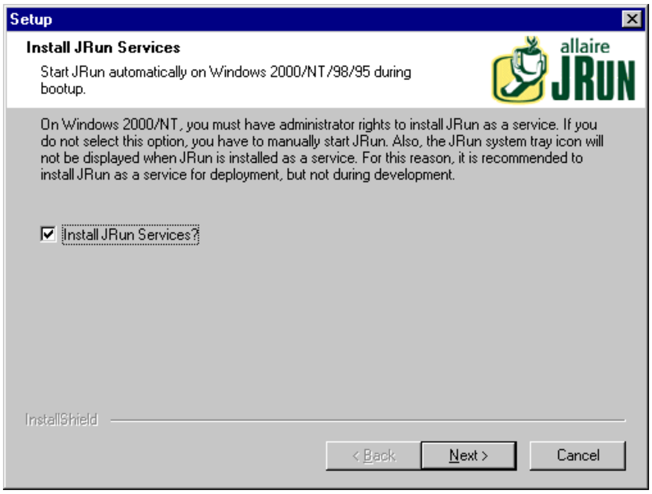
JRun 3.0

Existing Folders:

ActivePerl
Administrative Tools (Common)
Adobe
Apache JServ 1.1
Bright Tiger
ClusterCATS
IDA Pro 4.01 Demo
InstallShield 5.5 Professional Edition
JRun 3.0

11. Select a program folder name for JRun and click Next.

    JRun installs the requested files. The Install JRun Services window appears.


Setup

Install JRun Services

Start JRun automatically on Windows 2000/NT/98/95 during bootup.

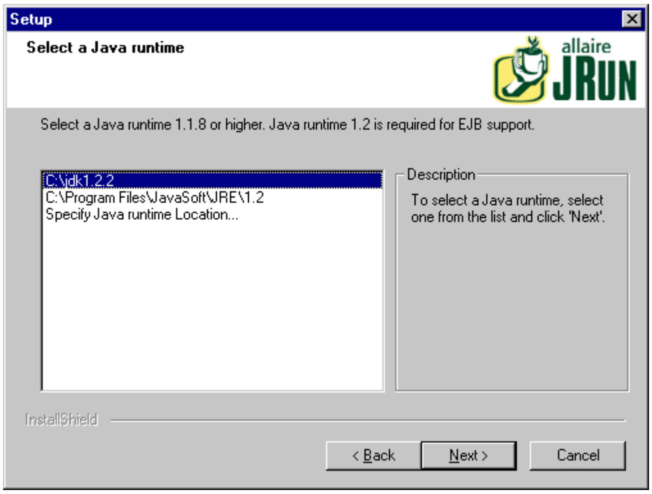
On Windows 2000/NT, you must have administrator rights to install JRun as a service. If you do not select this option, you have to manually start JRun. Also, the JRun system tray icon will not be displayed when JRun is installed as a service. For this reason, it is recommended to install JRun as a service for deployment, but not during development.

☑ Install JRun Services?

12. Choose whether or not to have the JRun servers start automatically and click Next.

    If you select Install JRun Services, the JRun Admin Server and JRun Default Server start up every time the machine boots. On NT, they are installed as NT services, which run as system processes rather than as user processes. On Windows 95/98, the servers are referenced in the Windows registry and automatically start on reboot. If you do not select Install JRun Services, the JRun servers run as applications and must be started manually.

The Select a Java Runtime window appears.



Select a Java runtime

Select a Java runtime 1.1.8 or higher. Java runtime 1.2 is required for EJB support.

C:\jdk1.2.2
C:\Program Files\JavaSoft\JRE\1.2
Specify Java runtime Location...

Description

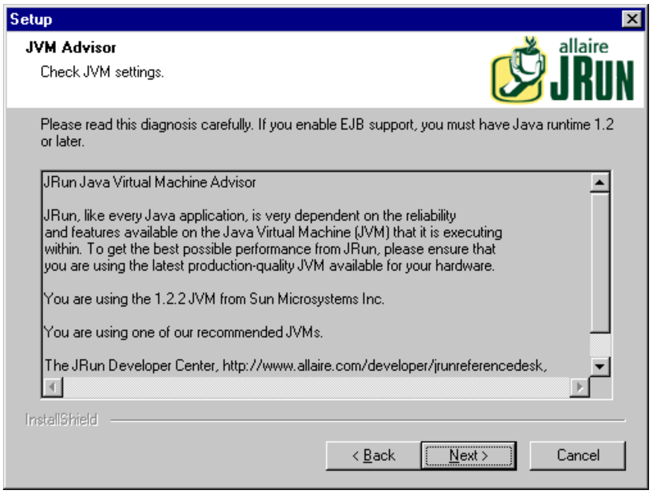
To select a Java runtime, select one from the list and click 'Next'.

13. Select a runtime environment for Java and click Next.

Note To use JRun's Enterprise JavaBeans components you must select a JRE version 1.2.2 or later.

Since Sun's Java Runtime Environment (JRE) is included with the Windows version of JRun, you are not required to provide your own JRE. To install it, cancel the installation and click the "Java Runtime Environment 1.2.2" link on the JRun splash screen. After installing the JRE, begin this installation procedure again.

The JVM Advisor appears.



JVM Advisor

Check JVM settings.

Please read this diagnosis carefully. If you enable EJB support, you must have Java runtime 1.2 or later.

JRun Java Virtual Machine Advisor

JRun, like every Java application, is very dependent on the reliability and features available on the Java Virtual Machine (JVM) that it is executing within. To get the best possible performance from JRun, please ensure that you are using the latest production-quality JVM available for your hardware.

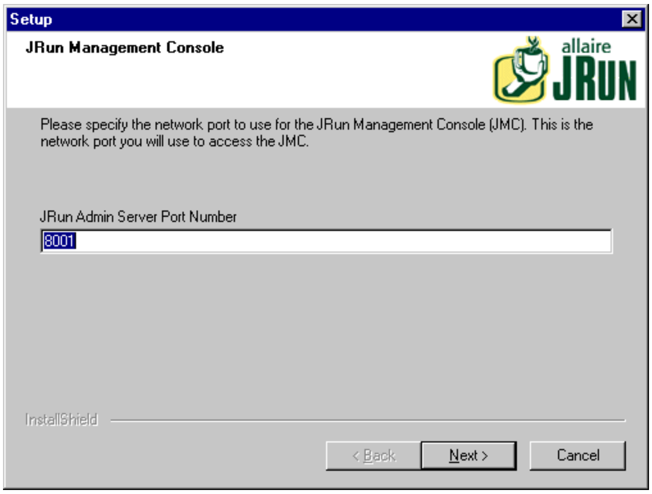
You are using the 1.2.2 JVM from Sun Microsystems Inc.

You are using one of our recommended JVMs.

The JRun Developer Center, http://www.allaire.com/developer/jrunreferencedesk,

14. Verify that the information in the JVM Advisor is acceptable and click Next.

    The JRun Management Console Admin Port window appears.



Please specify the network port to use for the JRun Management Console (JMC). This is the network port you will use to access the JMC.

JRun Admin Server Port Number

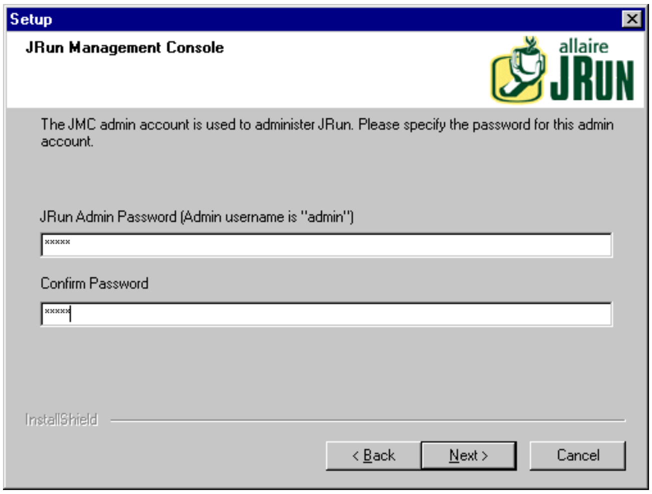
8001

15. Enter a unique port number that you will use to access JRun's administrative Web application on the JRun Web Server and click Next.

    The JRun Web Server (JWS) listens on this port to provide access to the JRun Management Console (JMC). The default port number is 8000.

    Note Do not select a port number between 8100 and 8199. JRun uses a port in this range for the `default` JRun server's JWS.

    The JRun Management Console Password window appears.



The JMC admin account is used to administer JRun. Please specify the password for this admin account.

JRun Admin Password (Admin username is "admin")

*****

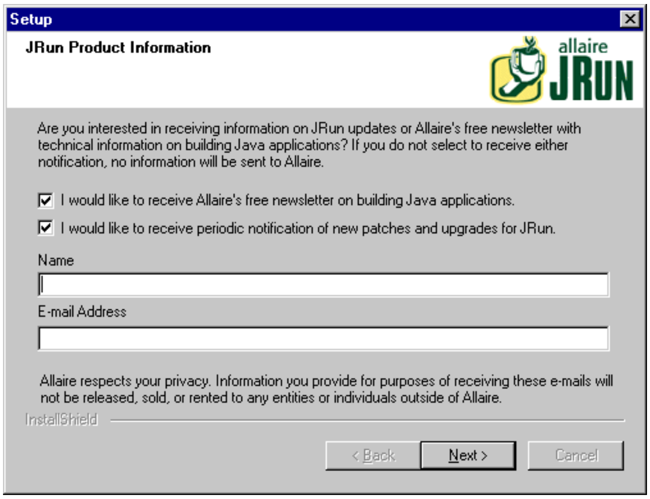
Confirm Password

*****

16. Enter and confirm the password for the JRun administrator (`admin`) and click Next.

Note Do not use spaces or the asterisk (*) character in the password.

After the JRun installer creates the appropriate directories and extracts the system files, the Product Information window appears.



Are you interested in receiving information on JRun updates or Allaire's free newsletter with technical information on building Java applications? If you do not select to receive either notification, no information will be sent to Allaire.

☑ I would like to receive Allaire's free newsletter on building Java applications.

☑ I would like to receive periodic notification of new patches and upgrades for JRun.

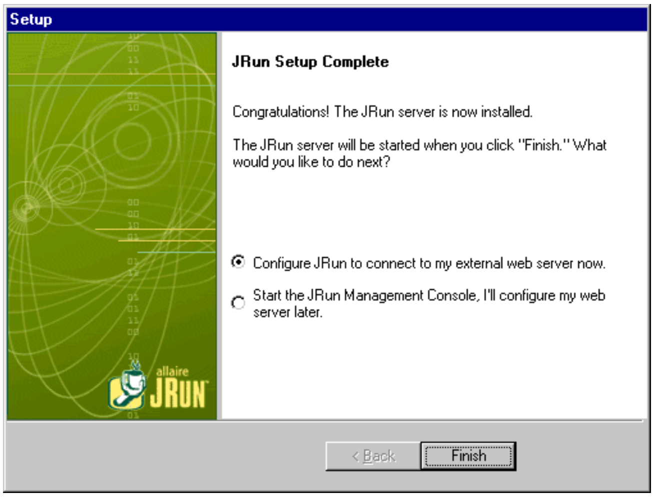
Name

E-mail Address

Allaire respects your privacy. Information you provide for purposes of receiving these e-mails will not be released, sold, or rented to any entities or individuals outside of Allaire.

17. Optionally, enter your name and e-mail address in the fields provided. Select the checkboxes to receive information about Java application developments and/or notifications about JRun and click Next.

The Setup Complete window appears.



JRun Setup Complete

Congratulations! The JRun server is now installed.

The JRun server will be started when you click "Finish." What would you like to do next?

[✓ Configure JRun to connect to my external web server now.]

Start the JRun Management Console, I'll configure my web server later.

18. To configure the connection between JRun and your external Web server (such as Apache or IIS), select the first radio button and click Finish. JRun launches the JRun Management Console and prompts you to log in. Once you log in, JRun brings you to the Connector Wizard.

    To configure your external Web server later, select the second radio button and click Finish. JRun opens the JRun Management Console.

    For instructions on logging in and finishing the configuration, refer to “Launching the JRun Management Console” on page 27.

UNIX and Linux installation

This section describes how to install JRun on a UNIX/Linux system.

To install JRun:

1. Stop your Web server. You must do this if you are going to connect JRun to your Web server.
2. Make sure you have already installed the desired Java Runtime Environment (JRE) on your machine. A JRE 1.6 or higher is required for JSP/servlets, and JRE 1.2 or higher is required for EJB support. You can obtain Sun’s JRE from the following Web site:

   `http://java.sun.com`
3. Set execute permission for the `jr30[n].sh` file, the JRun installation shell script, using the following command:

   ```
   chmod 755 jr30[n].sh
   ```

   The name of the script depends on your flavor of UNIX:

jr30h.sh	HP/UX
jr30l.sh	Linux
jr30s.sh	Solaris
jr30x.sh	Irix
jr30i.sh	AIX
jr30o.sh	Tru64
jr30g.sh	Generic UNIX platforms

4. Execute the JRun installation script using the following command:

   ```
       /bin/sh ./jr30[n].sh
   ```

   JRun asks you to read the license agreement.
5. Press Enter to view each page of the license agreement.

JRun prompts you to accept the license agreement.

6. Enter `y` to accept the agreement or `n` to cancel the installation.

   JRun prompts you to enter the installation directory.

7. Enter a directory to install JRun into. This directory will be referred to as the <*`jrun_directory`*> in this document. The default is `/opt/JRun`.

   JRun prompts you to select the type of installation to perform. You can select either Typical or Custom.

   If you select Typical, JRun installs all components. If you select Custom, you are prompted with the following options:

   ```
   1. Servlet and Java ServerPages
   2. Enterprise Java Beans and Java Message Service
   3. All
   ```

8. Enter the type of installation.

   JRun unpacks and copies all the files necessary for your installation and then prompts you to enter the absolute path to your JRE or JDK directory.

9. Specify the location of your JRE/JDK. Typically, the JRE/JDK is installed in `/usr/java`, but it may be in a different location on your system.

   Note If you select a JRE/JDK prior to 1.2, JRun's Enterprise JavaBeans components will not work correctly.

   JRun prompts you to enter your license key.

10. Enter the license key *exactly* as it was provided to you by Allaire.

    If you are installing the JRun Developer Edition or an evaluation version of JRun, leave the entry blank (the default). If you are upgrading from a previous version of JRun, enter your new upgrade key. You will then be prompted for your old 2.*x* license key.

    JRun prompts you to enter a password for the JRun administrator.

11. Enter a password. Do not use spaces or the asterisk (*) character in the password.

    JRun prompts you to enter a port number.

12. Enter a unique port number that you will use to access JRun's administrative Web application on the JRun Web Server. The JRun Web Server (JWS) listens on this port to provide access to the JRun Management Console (JMC). The default port number is 8000.

    Note Do not enter a port number between 8100 and 8199. JRun uses a port in this range for the `default` JRun server's JWS.

    JRun prompts you to receive information about Java application developments and/or notifications about JRun.

13. Choose whether or not to receive this information.

    If you entered `Yes`, JRun then prompts you for your name and e-mail address.

14. Enter your name and e-mail address.

    JRun prompts you to either open the JMC URL or the demo URL in your browser.

    To continue with the configuration and connect JRun to your external Web server, open the URL for the JRun Management Console (JMC). For instructions on finishing the configuration, refer to "Launching the JRun Management Console" on page 27.

    You can launch the JMC at any time to configure your JRun implementation. For more information, refer to "Starting the JRun Management Console" on page 78.

Launching the JRun Management Console

The JRun Management Console (JMC) is a Web application that gives you a browser-based interface for configuring JRun. To use the JMC, you must have either Netscape Communicator Version 4.0 or later or Internet Explorer Version 4.0 or later.

Note This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

To launch the JMC:

1. You can start the JMC in UNIX and Windows by:

   - Opening the following URL in your Web browser:

     ```
     http://localhost:8000
     ```

   Or **Windows only** (any one of the following):

   - Selecting `Start > Programs > JRun 3.0 > JRun Management Console`
   - Double-clicking the JRun icon in the system tray and clicking Start (if you installed JRun as an application).
   - Entering the following DOS command (in the *`<JRun_directory>`*`/bin` directory):

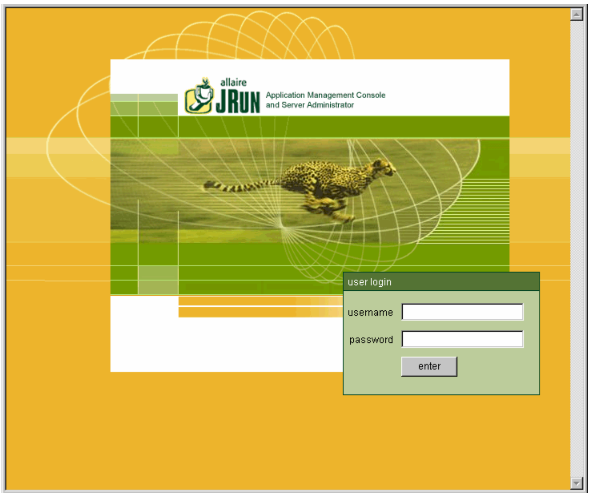
     ```
     jrun -admin
     ```

   If this is the first time you are launching the JMC, the JRun license agreement appears.

   If the JMC does not open, refer to "Troubleshooting" on page 35. For more information on using the JRun command-line options, refer to Chapter 3.

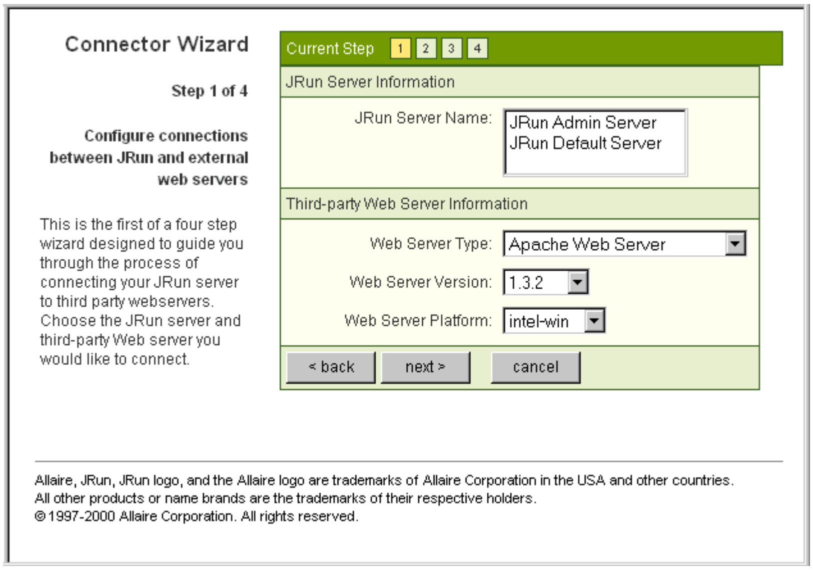
2. Accept the JRun license agreement. (You only have to accept the license agreement once.)

The JMC login window appears:



3. Enter your user name and password in the fields provided and click Login. The default user name is `admin`. You created the password for `admin` during the installation procedure.

   If you are launching the JMC as part of the installation procedure, the JRun Connector Wizard appears:



Configure connections between JRun and external web servers

This is the first of a four step wizard designed to guide you through the process of connecting your JRun server to third party webservers. Choose the JRun server and third-party Web server you would like to connect.

JRun Server Information

JRun Server Name: JRun Admin Server, JRun Default Server

Third-party Web Server Information

Web Server Type: Apache Web Server

Web Server Version: 1.3.2

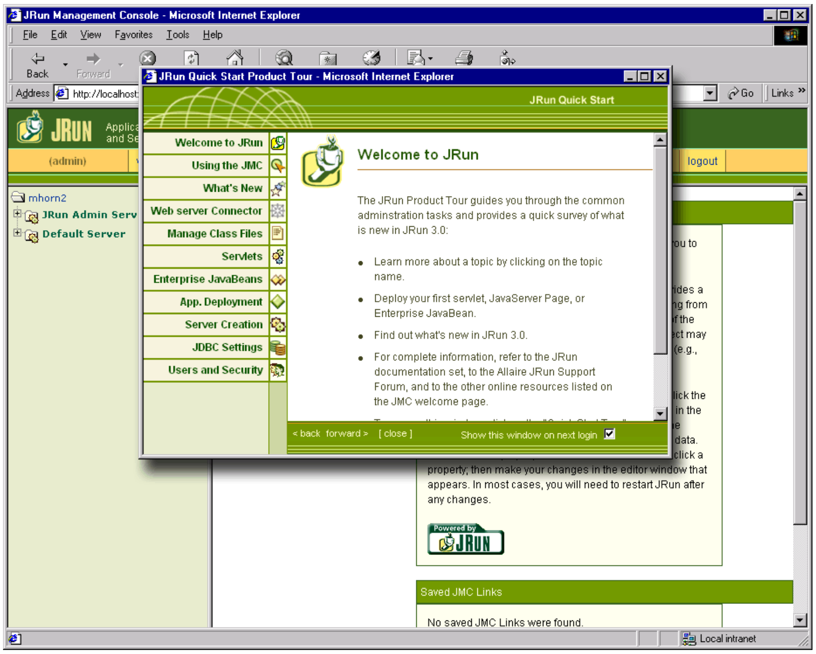
Web Server Platform: intel-win

Allaire, JRun, JRun logo, and the Allaire logo are trademarks of Allaire Corporation in the USA and other countries. All other products or name brands are the trademarks of their respective holders.
© 1997-2000 Allaire Corporation. All rights reserved.

To finish the setup, refer to your Web server’s configuration section in Chapter 2:

- “Configuring Apache” on page 38
- “Configuring IIS 3.0/PWS” on page 43
- “Configuring IIS 4.0/5.0” on page 46
- “Configuring Netscape/iPlanet” on page 53
- “Configuring WebSite Pro” on page 61
- “Configuring Java-Based Web Servers” on page 68
- “Configuring the Zeus Web Server” on page 70

If you are launching the JMC after installation, the JMC main window appears, with the JRun Quick Start Product Tour window:



Welcome to JRun

The JRun Product Tour guides you through the common adminstration tasks and provides a quick survey of what is new in JRun 3.0:

- Learn more about a topic by clicking on the topic name.
- Deploy your first servlet, JavaServer Page, or Enterprise JavaBean.
- Find out what's new in JRun 3.0.
- For complete information, refer to the JRun documentation set, to the Allaire JRun Support Forum, and to the other online resources listed on the JMC welcome page.

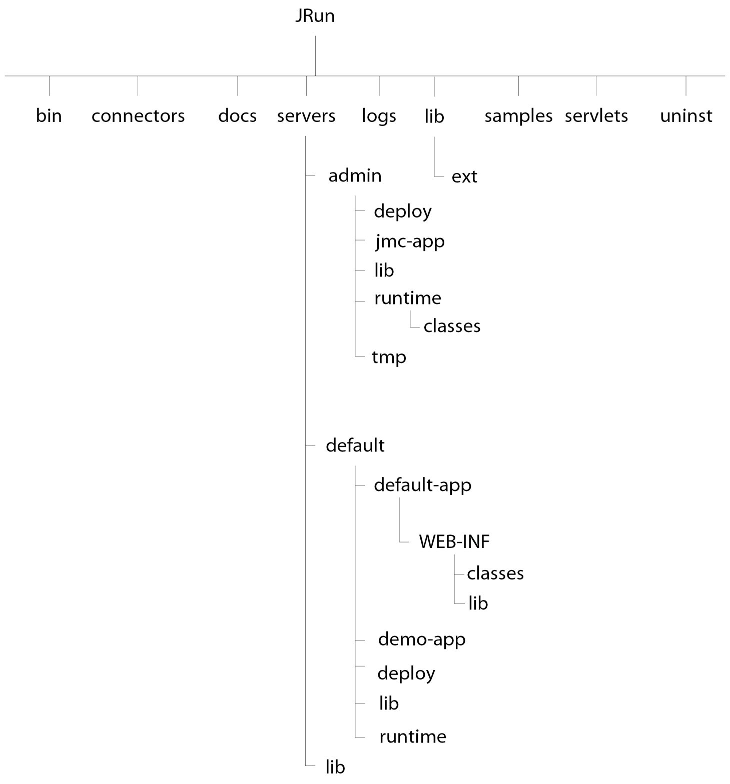
property; then make your changes in the editor window that appears. In most cases, you will need to restart JRun after any changes.

No saved JMC Links were found.

For information on using the JMC to configure JRun, refer to Chapter 3.

JRun Directory Structure

The following figure shows the JRun directory structure:



By default, JRun creates the `/JRun` directory under `c:\Program Files\Allaire` (Windows) and `/opt` (UNIX/Linux).

The contents of `/JRun` are described in the following table. Note that not all subdirectories appear in all directories since each implementation may vary.

JRun subdirectories

The following table describes the subdirectories in the `/JRun` directory.

Directory	Description
`/bin`	Contains JRun executable files.
`/connectors`	Contains Web server connector files.

Directory	Description
`/docs`	Contains HTML documentation for JRun and for the Java Servlet API.
`/lib`	Contains the JRun `.jar` files and properties files that define default properties for all JRun applications.
`/lib/ext`	Contains `.jar` files including `servlet.jar` and `ejb.jar`.
`/logs`	Contains the JRun log files.
`/samples`	Contains JRun sample files.
`/servers`	Contains the JRun servers and their applications.
`/servers/lib`	Contains `.jar` and `.class` files accessed by all JRun servers. This is a good location to store shared database drivers and other shared files. JRun stores the tag library in this directory.
`/servlets`	Contains `.class` files that are accessible to the default Web application. This directory is included for backwards compatibility. The `.class` files for a new application should be arranged in a structured hierarchy as defined by the Servlet 2.2 specification.
`/uninst`	Contains JRun uninstallation information.

Admin JRun server subdirectories

The following table describes the subdirectories in the `/servers/admin` directory.

Directory	Description
`/servers/admin`	Defines the administration JRun server.
`/servers/admin/deploy`	Stores Enterprise JavaBeans (EJBs) to be deployed. Once deployed, EJBs are copied to the `runtime` directory at startup.
`/servers/admin/jmc-app`	Contains the JRun Management Console (JMC) application.
`/servers/admin/lib`	Contains `.jar` and `.class` files accessed by all applications within the `admin` server.
`/servers/admin/runtime`	EJB runtime directory.
`/servers/admin/tmp`	Contains temporary subdirectories for each application in this JRun server. *Do not remove these temporary directories.*

Default JRun server subdirectories

The following table describes the subdirectories in the `/servers/default` directory. If you create a new JRun server, these subdirectories will be part of that server

Directory	Description
`/servers/default`	Defines the default JRun server.
`/servers/default/default-app`	Contains the default JRun application. You use this application to build and test Java servlets and JSPs.
`/servers/default/default-app/WEB-INF`	Contains all resources related to the `default` application that are not in the document root. Note that this directory is not part of the document tree of the application. That is, no file contained in this directory may be served directly to a client. This directory contains the application descriptor `web.xml`.
`/servers/default/default-app/WEB-INF/classes`	Contains the Java class files for the Web application's servlets.
`/servers/default/default-app/WEB-INF/jsp`	Contains `.class` files for the application's JSPs.
`/servers/default/default-app/WEB-INF/lib`	Contains beans and other files used by the application. These files may be contained within `.jar` files.
`/servers/default/demo-app`	Contains the JSP/servlet sample applications.
`/servers/default/deploy`	Stores deployed Enterprise JavaBeans (EJBs). Deployed EJBs are copied to the `runtime` directory at startup.
`/servers/default/lib`	Contains `.jar` and `.class` files accessed by all applications within the `default` server.
`/servers/default/runtime`	Deployed EJBs are copied to the `runtime` directory at startup.
`/servers/default/runtime/classes`	Contains class files for dynamically loaded EJB implementations.
`/servers/default/tmp`	Contains temporary subdirectories for each application in this JRun server. *Do not remove the temporary directories.*

Using JRun Servers

JRun provides utilities to start, stop and perform other functions on the JRun servers. These utilities are described below for the different JRun platforms. For more information about JRun servers, refer to “Configuring JRun Servers” on page 84.

Windows Considerations

During installation you can configure JRun to run as an NT service on Windows systems. If you do, JRun starts every time you start your NT system unless you disable the service. You can also use the Services Control Manager utility (found in the Control Panel) to perform the actions described in this section. If you do not run JRun as a service, it runs as an application.

The procedures described in this section for Windows can also be performed using the scriptable JRun command-line utility. For more information, refer to “Using the JRun command” on page 86.

Starting and Stopping JRun Servers

To start a JRun server, do one of the following:

- Windows:

  Select `Start > Programs > JRun 3.0 >` *`JRun_server_name`*. For example, to start the JRun admin server, select `Start > Programs > JRun 3.0 > JRun Admin Server`.

  Or

  Use the JRun command-line utility:
  ```
  jrun -start <JRun_server_name>
  ```

Note If you installed JRun as a service and try to launch JRun from the program group (by selecting `Start > Programs > JRun 3.0`) when the service is already running, you will get a "*JRun exited abnormally*" error.

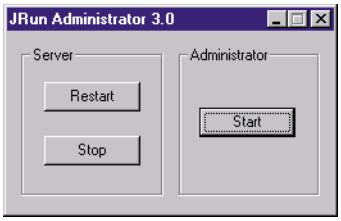
- UNIX:

  Use the JRun command-line utility:
  ```
  jrun -nohup -start <JRun_server_name>
  ```
  The `nohup` option starts the JRun server as a background process.

To stop a JRun server, do one of the following:

- Windows:

  If you are running the JRun server as an application, open the JRun Application Manager by double clicking the JRun server’s icon in your system tray:



Then click the Stop button. JRun stops only that JRun server.

If you are running JRun as an NT service, use the Services Control Manager utility (found in the Control Panel) to stop the JRun server.

- UNIX/Windows:

  Use the JRun command-line utility:
  ```
  jrun -stop <JRun_server_name>
  ```

To restart a JRun server, do one of the following:

- Windows:

  If you are running the JRun server as an application, open the JRun Application Manager by double clicking the JRun server's icon in your system tray. Then click the Restart button. JRun restarts only that JRun server.

  If you are running the JRun server as an NT service, use the Services Control Manager utility (found in the Control Panel) to stop and then start the JRun server.

- UNIX/Windows:

  Use the JRun command-line utility:
  ```
  jrun -restart <JRun_server_name>
  ```

  Or

  Click the *`machine_name`* in the left pane of the JRun Management Console (JMC). In the right pane, click the Restart link next to the JRun server you want to restart. JRun restarts the server. For more information on using the JMC, refer to "Launching the JRun Management Console" on page 27.

Note You cannot restart the `admin` JRun server using the JMC.

Starting the JRun Demo Application

JRun ships with sample EJBs, Java servlets, JavaServer Pages (JSPs), and a sample tag library accessible in the demo application. This section describes how to start the JRun demonstration on Windows and UNIX.

Note This procedure assumes that your `default` JRun server is running on the JRun-supplied Web server on the default port (8100). The JWS associated with the JMC runs on port 8000 by default.

For more information on the servlet, tag library, JSP, and EJB samples, refer to the *JRun Samples Guide*.

To start the JRun demo application:

1. Start the `default` JRun server if it is not already running using the instructions in "Starting and Stopping JRun Servers" on page 33.
2. Open the following URL in your Web browser:

   `http://localhost:8100/demo/index.html`

   Or (Windows only)

   Select `Start > Programs > JRun 3.0 > JRun Demo`.

Troubleshooting

The information in this section is designed to help you get past the most common problems associated with JRun installation.

When troubleshooting JRun, you can also check the log files located in <*`jrun_directory`*>`/logs` for additional information.

If the login page does not open correctly:

- Check the port number in the request URL. You may have overridden JRun's default (8000) during installation. If you did, use that value. If you are unsure, check the value of `web.endpoint.main.port` in the `Web Services` section of the <*`JRun_directory`*>`/servers/admin/local.properties` file.
- Restart your JRun servers after making any changes to the properties files.
- Make sure the `admin` JRun server is running:
  - In Windows, check the system tray. The JRun Admin Server icon should appear if you installed JRun as an application. If you installed JRun as a service, check the Services Control Manager utility (found in the Control Panel) to see if the `JRun Admin Server` service is running.
  - On UNIX, use the command-line tool in `/opt/jrun/bin` to determine if the server is running:

    ```
    jrun -status admin
    ```
- Make sure you have read access for the `/JRun/servers/admin` directory and its subdirectories.
- Make sure that the JMC is listening on port 8000. If you installed JRun 3.0 over a previous version, JRun may have incremented the default port number to 8001

after encountering a stray process. For more information, refer to “Installing JRun” on page 16.

If the demo application does not open:

- Be sure you specify the correct port when trying to access the `demo` application. It is running on the `default` JRun server (port 8100) and not the `admin` JRun server (port 8000):

  ```
  http://localhost:8100/demo/index.html
  ```

  The default port for the `default` server is 8100. However, if JRun detected another process using that port number during installation, it would have incremented the port number until a free one was found.

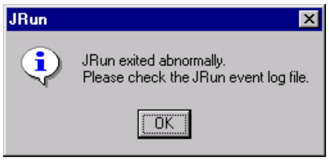
  You can see which port the default server is running on in the JRun Web Server panel of the JRun Management Console. For more information, refer to “Configuring the JWS” on page 101.

- Make sure the `default` JRun server is running:
  - In Windows, check the system tray. The Default Server icon should appear if you installed JRun as an application. If you installed JRun as a service, check the Services Control Manager utility (found in the Control Panel) to see if the `JRun Default Server` service is running.
  - On UNIX, use the command-line tool in `/opt/jrun/bin` to determine if the server is running:

    ```
    jrun -status default
    ```

- Try launching the demo application from the JMC by clicking the Example Applications link on the Welcome page.

If you get the following error when trying to start a JRun server in Windows:


JRun

JRun exited abnormally.
Please check the JRun event log file.

Make sure that JRun is not already running as a service if you are trying to launch the JRun server as an application. JRun services are invoked through the Services Control Manager utility (found in the Control Panel), whereas JRun applications are typically launched from the program group or the `Start` menu.

To tell if a JRun server is already running, check your toolbar for JRun icons. There should be one icon for each running server if they are running as applications. Also check the Services Control Manager utility if they are running as services and look for "JRun Admin Server" and "JRun Default Server."

CHAPTER 2

External Web Server Configuration

This chapter describes how to configure JRun to communicate with your Web server and the changes that JRun makes to that server's configuration. As part of the procedure, you may need to set configuration parameters for your Web server, then use the JRun Management Console (JMC) to configure the connection between JRun and the Web server.

See the appropriate section for instructions on configuring JRun with your specific Web server.

Note It is not necessary to have a separate Web server to develop applications using JRun. JRun provides its own Web server that is configured for you when you install JRun.

Contents

Configuration Overview

JRun 3.0 functions as both a stand-alone Java application server or a plug-in module that adds Java application support to an existing Web server.

Note If you are not connecting JRun as a plug-in with an external Web server, you can skip this chapter.

JRun supports a wide variety of Web servers. While the basic procedure for configuring the connection between JRun and a Web server is the same for all Web servers, each Web server has unique configuration information and settings. Below is a general process for configuring JRun with a Web server.

To configure the connection between JRun and any Web server:

1. Stop the Web server.
2. Configure the Web server to communicate with JRun (if necessary).
3. Start the JRun Management Console (JMC).
4. Run the JRun Connector Wizard to create a JRun Connection Module (JCM) that manages the connection between the Web server and the `default` JRun server.
5. Start the Web server and the default JRun server.
6. Verify the connection between JRun and the Web server.

The remaining sections describe this procedure for specific Web servers supported by JRun.

The connection between JRun and an external Web server takes the form of the JRun connector. Allaire provides a native connector for most major Web servers, but JRun also includes connector source code for use with unsupported Web servers. You can find this source code, along with basic usage instructions, in `<JRun_directory>/connectors/src`. For more information, refer to the *JRun Advanced Configuration Guide*, available from the Allaire Web site.

Configuring Apache

This section describes how to configure JRun to communicate with an Apache Web server running under Windows or UNIX.

Based on your operating system, JRun can support two methods for running servlets with an Apache Web server: the Dynamic Shared Objects (DSO) module and the static module. As part of configuring JRun to communicate with Apache, you may need to compile Apache for your specific module.

For Windows-based systems, Apache only supports the DSO module; you do not need to perform any configuration steps to set up the DSO module.

For UNIX-based systems, including Linux, DSO is recommended for Apache version 1.3.x because it makes it easier to build Apache. In addition, some platforms (such as

RedHat Linux 5.2) provide Apache pre-built with DSO support; so Apache does not need to be recompiled.

On all UNIX-based versions of Apache, JRun can be compiled into the server as a static module; however, the static module is recommended only for systems that do not support DSO.

If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you on how to launch the Connector Wizard from within the JMC.

To connect Apache and JRun:

1. Stop your Web server.
2. **UNIX only**: Configure the DSO module (if necessary for your system).

   This procedure will work only with Apache 1.3.x under UNIX.

   - Execute the following Apache command to configure it for DSO:

     ```
     ./configure --prefix=/user/local/apache --enable-rule=SHARED_CORE \
     --enable-module=so
     ```

   - Recompile and install Apache using the `make` and then `make install` scripts.
3. **UNIX only**: Configure the static module (if necessary for your system).

   Only configure the static module if you are not using the DSO module.

   The procedures for configuring the static module differ for Apache 1.2 and Apache 1.3.x under UNIX.

   Apache 1.2:

   - Copy the JRun source files from `<`*`JRun_directory`*`>/connectors/apache/src` into your Apache `/src/modules/jrun` directory.
   - Add the following line to the Apache `Configuration` file in the `src` directory:

     ```
     Module jrun_module modules/jrun/libjrun.a
     ```

   - From the Apache `src` directory, run the `configure` script to create a new `Makefile`; then recompile and install Apache.

   Apache 1.3.x:

   - Execute the following Apache command to add the JRun library to your Apache server:

     ```
     ./configure --prefix=/user/local/apache \
     --activate-module=src/modules/jrun/libjrun.a
     ```

     Note that the `--prefix` entry and other entries may be different. The operative entry is the `--activate-module` entry.

   - Recompile and install Apache using the `make` and then `make install` scripts.
4. Start the JMC by opening the following URL in your Web browser:

   ```
   http://localhost:8000
   ```

Or (Windows only): Select `Start > Programs > JRun 3.0 > JRun Management Console`.

Note This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

5. Log in to the JMC as the JRun administrator.
6. Select `connector wizard` in the access bar.
7. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings		
Step	**Parameter**	**Description**
Step 1	JRun Server Name	Select the JRun server you want to connect to Apache. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `Apache Web Server` from the drop-down list.
	Web Server Version	Select Apache's version.
	Web Server Platform	Choose the platform on which Apache is running.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to Apache. Use the default value of `127.0.0.1`, unless Apache uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with Apache. Do not confuse this with Apache's HTTP port. This book uses `51000` in the examples, but you can select any open port.

JRun Connection Module Settings		
Step	**Parameter**	**Description**
Step 3	Apache conf directory	Specify the directory containing the configuration files `srm.conf` and `httpd.conf`. To use JRun's Directory Reader, click Browse.
	DSO support	**UNIX:** Select DSO support if you compiled the JRun module into your Apache server. **Windows:** Select DSO support.

8. After you have successfully installed the JRun connector, restart your Apache Web server and the `default` JRun server.

   If the `default` JRun server is not already running:

   - Windows:

     If you installed JRun as an application, select `Start > Programs > JRun 3.0 > JRun Default Server`.

     If you installed JRun as a service, open the Services Control Manager utility (found in the Control Panel) and start the "`JRun Default Server`" service or use the JRun command-line utility:

     ```
     jrun -start default
     ```

   - UNIX:

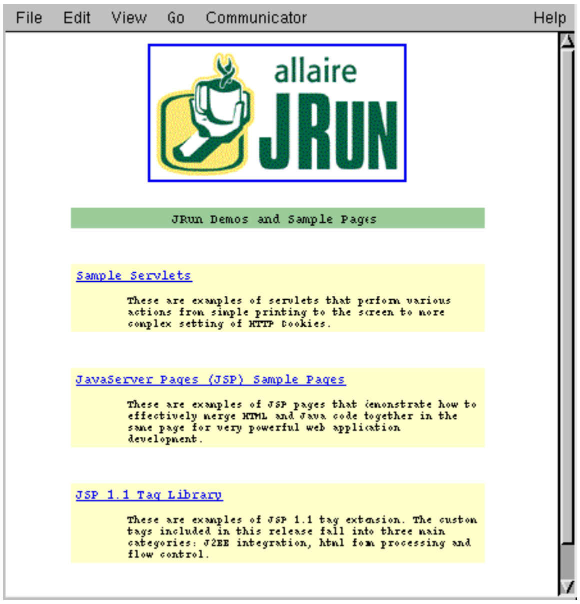
     Use the JRun command-line utility:

     ```
     jrun -start default
     ```

9. Verify the JRun connection to your Apache Web server by running the JRun demo application with the following URL:

   ```
   http://localhost:80/demo/index.html
   ```

   This assumes that Apache is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

If the demo application runs, you have successfully configured the connection between JRun and your Apache Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Changes to Apache configuration files

The Connector Wizard makes some changes to the Apache `httpd.conf` file by adding the `JRun Settings` section. The settings mirror the settings in the `jrun.ini` file which initializes the JRun DLL (on Windows systems). A typical `JRun Settings` section resembles the following:

```
# JRun Settings
# JRun - Comment out this line to disable DSO (ie you compiled module
# into your server.
LoadModule jrun_module "C:\Program Files\Allaire\JRun\connectors\
    apache\intel-win\mod_jrun136.dll"
<IfModule mod_jrun.c>
JRunConfig Verbose false
JRunConfig ProxyHost 127.0.0.1
JRunConfig ProxyPort 51000
JRunConfig Timeout 300
JRunConfig Mappings "C:\Program Files\Allaire\JRun\servers\
    default\local.properties"
</IfModule>
```

The Connector Wizard also changes the JRun `local.properties` files based on your input. For more information, refer to “Changes to local.properties” on page 72.

You should not have to modify any files for the JRun Connector Wizard to work. This information is presented for informational purposes only.

Configuring IIS 3.0/PWS

This section describes how to connect JRun to IIS 3.0 or Personal Web Server (PWS) on Windows 95/98/NT systems.

If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you on how to launch the Connector Wizard from within the JMC.

To connect JRun and IIS 3.0 or PWS:

1. Stop your Web server.

   Note For IIS 3.0, use the Web server utility in the Control Panel to stop your Web server. Do not use the Microsoft Management Console (MMC) to do this.

2. Start the JMC using one of the following methods:

   - Select `Start > Programs > JRun 3.0 > JRun Management Console`
   - Open the following URL in your Web browser:

     ```
     http://localhost:8000
     ```

   This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

3. Log in to the JMC as the JRun administrator.
4. Select `connector wizard` in the access bar.

5. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings

Connector Wizard Step	Parameter	Description
Step 1	JRun Server Name	Select the JRun server you want to connect to the Web server. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `Internet Information Server` or `Personal Web Server` from the drop-down list.
	Web Server Version	Select your Web server's version from the drop-down list.
	Web Server Platform	Select `intel-win` in the drop-down list.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to IIS/PWS. Use the default value of `127.0.0.1`, unless IIS/PWS uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with IIS/PWS. Do not confuse this with IIS/PWS' HTTP port. This book uses `51000` in the examples, but you can select any open port.
Step 3	[PWS's|IIS's] scripts Directory	Specify the location of IIS/PWS' `/scripts` directory. To use JRun's Directory Reader, click Browse.
	Install as a Global Filter	Select this checkbox to have JRun install an ISAPI filter to detect if HTTP requests are attempting to run servlets.

6. After you have successfully installed the JRun connector, restart IIS/PWS and the `default` JRun server.

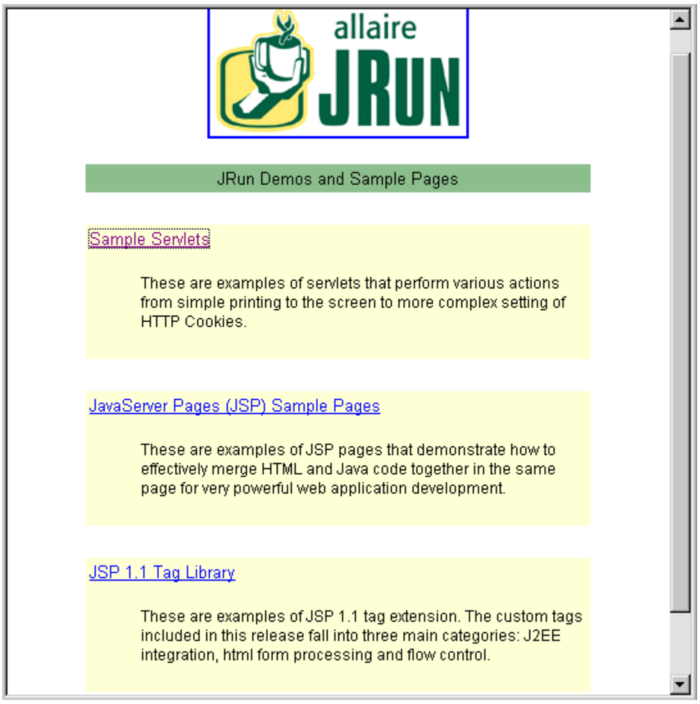
Note If you ran the connector for PWS, you must also reboot your computer.

If the `default` JRun server is not already running, select `Start > Programs > JRun 3.0 > JRun Default Server`.

7. Verify the JRun connection to your IIS/PWS Web server by running the JRun demo application with the following URL:

   ```
   http://localhost:80/demo/index.html
   ```

   This assumes that IIS/PWS is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

If the demo application runs, you have successfully configured the connection between JRun and your IIS/PWS Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Changes to configuration files

The Connector Wizard makes the following changes to your Web server implementation:

- Adds the `jrun.ini` and `jrun.dll` files in the `/inetpub/scripts` directory. JRun uses the `.ini` file to initialize the DLL on startup. The DLL is an ISAPI filter that intercepts HTTP requests to the Web server and passes the appropriate ones on to JRun for processing.
- Modifies the Windows registry. The Wizard adds the "`Filter DLLs`" key which points to the `jrun.dll`. The value of the key is the absolute path of the

`jrun.dll`. The key is located in `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/W3SVC/Parameters/`.

- If you selected `"Install as a global filter"`, the Connector Wizard adds a pointer to the JRun Connector Filter, which you can configure in the MMC. For more information, refer to "Configuring the JRun ISAPI filter" on page 51.
- Updates the JRun `local.properties` files. For more information, refer to "Changes to local.properties" on page 72.

You should not have to modify the registry or update any files for the JRun Connector Wizard to work. This information is presented for informational purposes only.

Configuring IIS 4.0/5.0

You can configure JRun to work with Internet Information Server 4.0 on Windows NT and 5.0 on Windows 2000 systems. You must give IIS execute permissions for the directories in which JSPs or servlets are going to run and then use the JRun Connector Wizard to connect JRun to IIS. This section describes the following:

- "Connecting JRun to IIS 4.0/5.0" on page 46
- "Changes to the IIS configuration files" on page 50
- "Configuring the JRun ISAPI filter" on page 51

If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you on how to launch the Connector Wizard from within the JMC.

Connecting JRun to IIS 4.0/5.0

To connect JRun and IIS 4.0/5.0:

1. Stop World Wide Web Publishing Service in the Services Control Manager utility (found in the Control Panel).

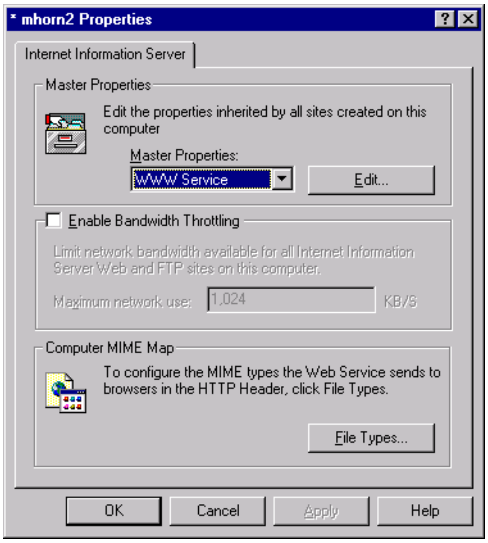
   Note On Windows NT, do not use the Web services Snap-In in the Microsoft Management Console (MMC) to do this.

2. Open the Internet Service Manager.

   On Windows NT: Select `Start > Programs > NT 4.0 Option Pack > Microsoft Internet Information Server > Internet Service Manager`. The Microsoft Management Console (MMC) appears and opens the `iis.msc` Snap-In.

   On Windows 2000: Select `Start > Programs > Administrative Tools > Internet Services Manager`. The Internet Services Manager appears.

3. Right click on either a specific virtual Web server or the entire Web server and select `Properties`. The Properties window appears.



Edit the properties inherited by all sites created on this computer

Master Properties: WWW Service

Enable Bandwidth Throttling

Limit network bandwidth available for all Internet Information Server Web and FTP sites on this computer.

Maximum network use: 1,024 KB/S

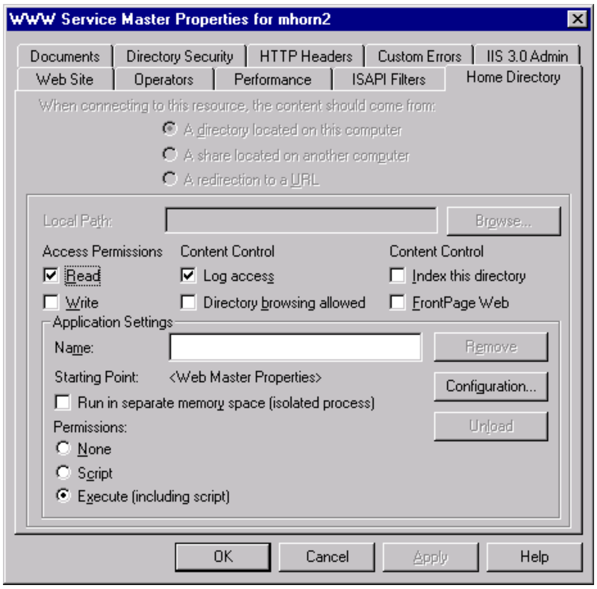
To configure the MIME types the Web Service sends to browsers in the HTTP Header, click File Types.

4. Select `WWW Service` from the `Master Properties` drop-down listbox and click Edit. The WWW Service Master Properties application appears.
5. Select the Home Directory tab.
6. Set execute permissions for the directory specified in the `Local Path` field.

   Windows NT: Under `Permissions`, select `Execute (including script)`.

   Windows 2000: In the `Execute Permissions` drop-down listbox, select `Scripts & Executables`.

   If there is no directory specified in `Local Path`, then you are setting a global property that affects all directories. You must make files in the `/scripts` directory executable.


WWW Service Master Properties for mhorn2
Documents | Directory Security | HTTP Headers | Custom Errors | IIS 3.0 Admin
Web Site | Operators | Performance | ISAPI Filters | Home Directory
Browse...
Remove
Configuration...
Unload
OK | Cancel | Apply | Help

When connecting to this resource, the content should come from:
[✓ A directory located on this computer]
A share located on another computer
A redirection to a URL

Local Path:

Access Permissions: [✓ Read], Write
Content Control: [✓ Log access], Directory browsing allowed
Content Control: Index this directory, FrontPage Web

Application Settings
Name:
Starting Point: <Web Master Properties>
Run in separate memory space (isolated process)
Permissions: None, Script, [✓ Execute (including script)]

7. Click OK to apply your changes. On Windows 2000, you may be prompted to change Inheritance Overrides. Click OK.

8. Start the JMC using one of the following methods:

   - Select `Start > Programs > JRun 3.0 > JRun Management Console.`
   - Open the following URL in your Web browser:

     `http://localhost:8000`

   This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

9. Log in to the JMC as the JRun administrator.

10. Select `connector wizard` in the access bar.

11. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings		
Connector Wizard Step	**Parameter**	**Description**
Step 1	JRun Server Name	Select the JRun server you want to connect to IIS. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `Internet Information Server` from the drop-down list.
	Web Server Version	Select `4.0` or `5.0` from the drop-down list.
	Web Server Platform	Select `intel-win` from the drop-down list.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to IIS. Use the default value of `127.0.0.1`, unless IIS uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with IIS. Do not confuse this with IIS' HTTP port. This book uses `51000` in the examples, but you can select any open port.
Step 3	IIS scripts Directory	Specify the location of IIS' `/scripts` directory. To use JRun's Directory Reader, click Browse.
	Install as a global filter	Select this checkbox to have JRun install an ISAPI filter to detect if HTTP requests are attempting to run servlets.

12. After you have successfully installed the JRun connector, restart your IIS Web server and the `default` JRun server.

    If the `default` JRun server is not already running:

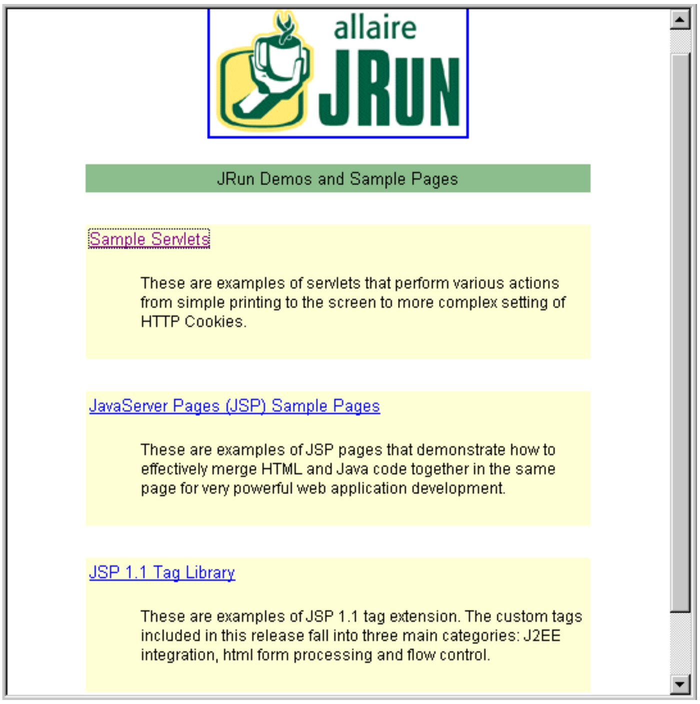
- If you installed JRun as an application, select `Start > Programs > JRun 3.0 > JRun Default Server`.
- If you installed JRun as a service, open the Services Control Manager utility (found in the Control Panel) and start the "`JRun Default Server`" service or use the JRun command-line utility:

```
jrun -start default
```

13. Verify the JRun connection to your IIS Web server by running the JRun demo application with the following URL:

```
http://localhost:80/demo/index.html
```

This assumes that IIS is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

If the demo application runs, you have successfully configured the connection between JRun and your IIS Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Changes to the IIS configuration files

The JRun Connector Wizard makes the following changes to your IIS implementation:

- Adds the `jrun.ini` and `jrun.dll` files in the `/inetpub/scripts` directory. JRun uses the `.ini` file to initialize the DLL on startup. The DLL is an ISAPI filter that intercepts HTTP requests to the Web server and passes the appropriate ones to JRun for processing.

- Maps `.jsp` to `jrun.dll` in IIS' Application Mappings settings.
- Makes several changes to the Internet Services Manager's metabase. The metabase is a hierarchical structure that stores IIS settings. JRun's changes to the metabase include:
  - Appending the `jrun.dll`'s absolute path to the `ScriptMaps` parameter. `ScriptMaps` maps filename extensions to DLLs for processing.
  - If you selected `"Install as a global filter"`, the Connector Wizard also adds the `JRun Connector Filter` service object and its associated parameters in the metabase `Filter` object.
- Updates the JRun `local.properties` files. For more information, refer to "Changes to local.properties" on page 72.

You should not have to modify the registry or metabase or edit any files for the JRun Connector Wizard to work. This information is presented for informational purposes only.

Configuring the JRun ISAPI filter

ISAPI filters can respond to events when IIS or PWS receives an HTTP request. If you select `"Install as a global filter"` while running the JRun Connector Wizard, JRun installs a DLL that is added to the other ISAPI filters in the Web server's memory. The default location of `jrun.dll` is `c:\inetpub\scripts\`.

The instructions in this section are optional. The default configuration of JRun does not require you to make any changes to the JRun ISAPI filter. However, if you install other ISAPI filters, you may need to make changes.

Editing the JRun ISAPI filter

Using the ISAPI Filters dialog box, you can add, remove, and change the name and location of the JRun ISAPI filter for IIS.

To edit the JRun ISAPI filter:

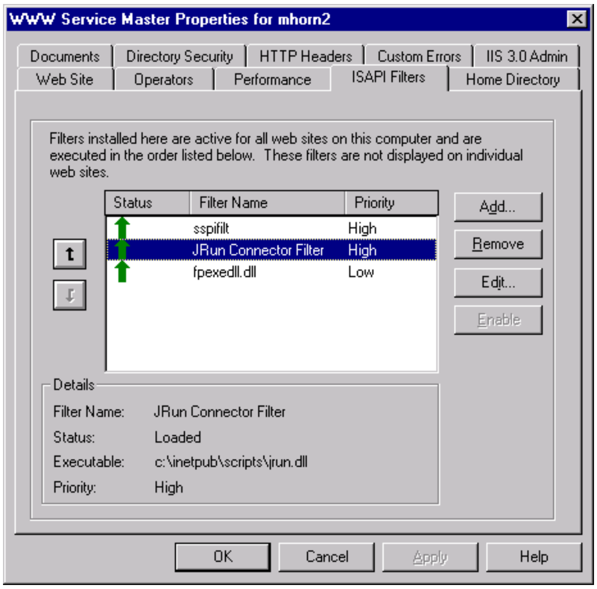
1. Open the Internet Service Manager.

   On Windows NT: Select `Start > Programs > NT 4.0 Option Pack > Microsoft Internet Information Server > Internet Service Manager`. The Microsoft Management Console (MMC) appears and opens the `iis.msc` Snap-In.

   On Windows 2000: Select `Start > Programs > Administrative Tools > Internet Services Manager`. The Internet Services Manager appears.

2. Right click on the machine name and select `Properties`. The Properties window appears.

3. Select `WWW Service` from the `Master Properties` drop-down listbox and click Edit. The WWW Service Master Properties application appears.
4. Select the ISAPI Filters tab.



Filters installed here are active for all web sites on this computer and are executed in the order listed below. These filters are not displayed on individual web sites.

Status	Filter Name	Priority
⬆	sspifilt	High
⬆	JRun Connector Filter	High
⬆	fpexedll.dll	Low

Details

Filter Name:	JRun Connector Filter
Status:	Loaded
Executable:	c:\inetpub\scripts\jrun.dll
Priority:	High

5. To remove the filter, select the JRun Connector Filter from the list of available ISAPI filters and click Remove.
6. To edit the name or location of the JRun filter DLL, select the JRun Connector Filter from the list of available ISAPI filters and click Edit.
7. To add a new JRun filter, click Add and browse for the location of the new filter.

   Note You should remove the old filter before adding a new one.

8. To apply your changes, click OK.
9. Restart your Web server.

Prioritizing the JRun ISAPI filter

When several ISAPI filters have registered for the same event (or notification), they are called sequentially by IIS. Filters with a higher priority are run before filters with a lower priority. The priority levels High, Medium, and Low are read-only properties that cannot be changed using the Internet Services Manager or other metabase editor. JRun's priority level is High.

While you cannot change a filter's priority level, you can determine which filter responds to an event first if it shares the same priority level as another filter. Use the procedure below to change the JRun ISAPI filter's priority.

To change the priority of the JRun ISAPI filter:

1. Open the Internet Service Manager.

   On Windows NT: Select `Start > Programs > NT 4.0 Option Pack > Microsoft Internet Information Server > Internet Service Manager`. The Microsoft Management Console (MMC) appears and opens the `iis.msc` Snap-In.

   On Windows 2000: Select `Start > Programs > Administrative Tools > Internet Services Manager`. The Internet Services Manager appears.

2. Right click on the machine name and select `Properties`. The Properties window appears.
3. Select `WWW Service` from the `Master Properties` drop-down listbox and click Edit. The WWW Service Master Properties application appears.
4. Select the ISAPI Filters tab.
5. Select `JRun Connector Filter` from the list of available ISAPI filters.
6. Click the up arrow to move the JRun filter ahead of the other filters in the list.
7. Click OK to apply your changes.
8. Restart you Web server.

Configuring Netscape/iPlanet

This section contains instructions for configuring the Netscape Web server.

Note As part of configuring Netscape, you may need to enable the Netscape Java Interpreter. However, you will not know this until after you have started the JRun Connector Wizard. For information on how to enable the Java interpreter, refer to "Enabling the Java interpreter" on page 56.

If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you on how to launch the Connector Wizard from within the JMC.

To connect JRun and Netscape Web servers:

1. Stop your Web server.
2. Start the JMC by opening the following URL in your Web browser:

   ```
   http://localhost:8000
   ```

   Or (Windows only):

   Select `Start > Programs > JRun 3.0 > JRun Management Console`.

   Note This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

3. Log in to the JMC as the JRun administrator.

4. Select `connector wizard` in the access bar.
5. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings

Connector Wizard Step	**Parameter**	**Description**
Step 1	JRun Server Name	Select the JRun server you want to connect to your Web server. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `Netscape Enterprise Server` or `Netscape FastTrack Server` from the drop-down list.
	Web Server Version	Select your Web server's version from the drop-down list.
	Web Server Platform	Choose the platform on which your Netscape Web server is running.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to NES. Use the default value of `127.0.0.1`, unless NES uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with NES. Do not confuse this with NES' HTTP port. This book uses `51000` in the examples, but you can select any open port.

JRun Connection Module Settings		
Connector Wizard Step	**Parameter**	**Description**
Step 3	Netscape http-xxx directory	Specify the `https` or `httpd` directory. This directory is usually named `httpd-xxx` or `https-xxx` under your `/Netscape/suitespot` directory. To use JRun's Directory Reader, click Browse.
	Native or Java Connector	Select either the Native (default) or Java connector for the Netscape Web server. The native connector uses NSAPI to communicate with the Web server and is recommended over the Java connector. The Java connector is an all-Java connector that implements Netscape's Server Applet API. Use this connector only if a native connector for your platform is not available. Choosing this connector requires that Java be enabled for your Netscape server prior to installing the connector. For information on enabling the Java interpreter, refer to "Enabling the Java interpreter" on page 56.

6. After you have successfully installed the JRun connector, restart your NES/iPlanet Web server and the `default` JRun server.

   If the `default` JRun server is not already running:

   - Windows:

     If you installed JRun as an application, select `Start > Programs > JRun 3.0 > JRun Default Server`.

     If you installed JRun as a service, open the Services Control Manager utility (found in the Control Panel) and start the "`JRun Default Server`" service or use the JRun command-line utility:

     ```
     jrun -start default
     ```

   - UNIX:

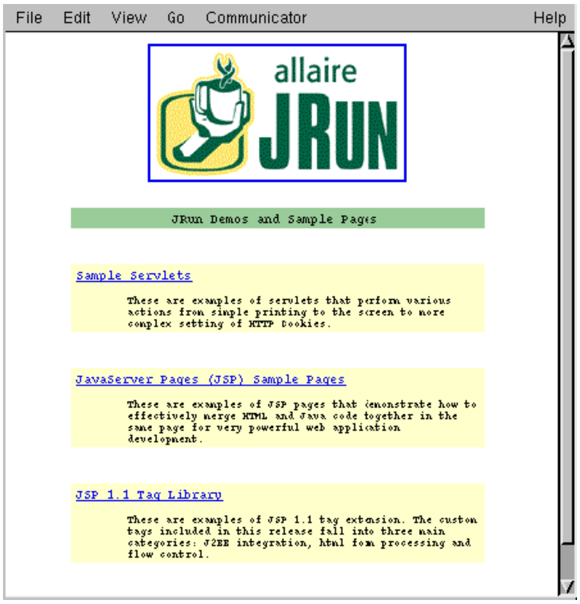
     Use the JRun command-line utility:

     ```
     jrun -start default
     ```

7. Verify the JRun connection to your NES/iPlanet Web server by running the JRun demo application with the following URL:

   ```
   http://localhost:80/demo/index.html
   ```

   This assumes that NES/iPlanet is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

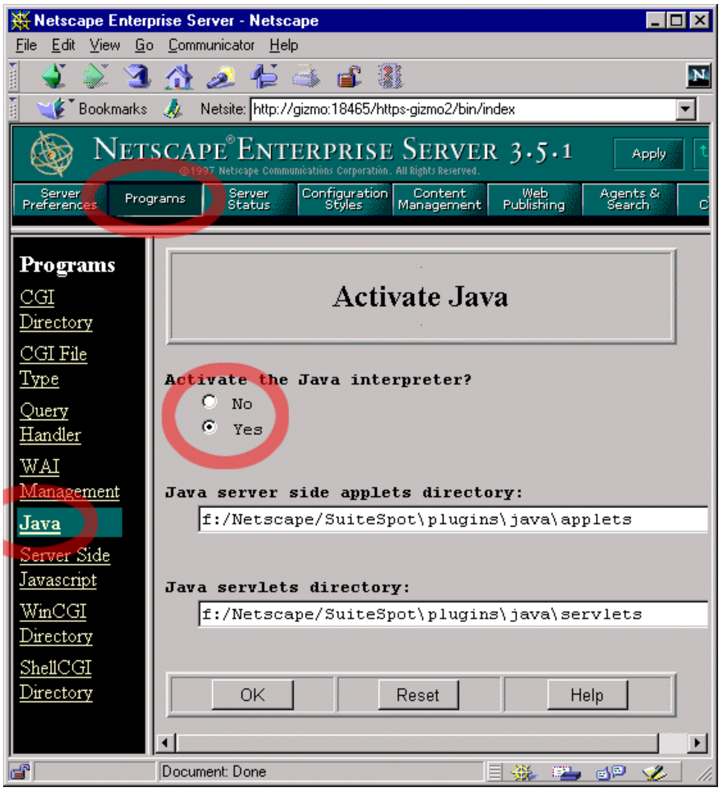
If the demo application runs, you have successfully configured the connection between JRun and your NES/iPlanet Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Enabling the Java interpreter

Native connectors are available for most platforms. The Netscape Web server requires no additional configuration steps unless a native connector is *not* available. In this case, you must enable the Java Interpreter built into the Netscape Web server before adding the JRun connector. This section describes how to enable the Java interpreter for Netscape 3.5.*x*.

To enable the Java interpreter for NES 3.5.*x*:

1. Start your Netscape Enterprise Administration Server and select the `Programs` menu.



Activate Java

Activate the Java interpreter? No [✓ Yes]

Java server side applets directory: f:/Netscape/SuiteSpot\plugins\java\applets

Java servlets directory: f:/Netscape/SuiteSpot\plugins\java\servlets

2. Under `Programs`, select `Java`.
3. Under `Activate the Java interpreter?`, select `Yes`.
4. Use the JRun Connector Wizard to configure the connection between JRun and Netscape. This procedure is described in “Configuring Netscape/iPlanet” on page 53.

Changes to NES configuration files

The JRun Connector Wizard modifies the `obj.conf` file in the `Netscape/Server4/https-<`*`machine_name`*`>/config/` directory. These changes, detailed below, depend on whether you are using a native or Java connector.

The Connector Wizard also changes the JRun `local.properties` files. For more information, refer to “Changes to local.properties” on page 72.

You should not have to modify any files for the JRun Connector Wizard to work. This information is presented for informational purposes only.

If you do modify the `obj.conf` file, note that the `proxyhost` settings must be an IP address and not a server name. A sample `obj.conf` file is provided on page 60. For more information on editing the `obj.conf` file, refer to the Netscape documentation.

Native connector (most common)

JRun makes the following changes to the `obj.conf` file when connecting to Netscape's Web server using a native connector.

- Adds initialization of the JRun NSAPI filter and sets the initialization parameters. The NSAPI filter intercepts HTTP requests to the Web server and passes the appropriate ones on to JRun for processing. A typical initialization in the `obj.conf` file resembles the following:

  ```
  Init fn="load-modules" shlib="C:/JRun/connectors/nsapi/intel-win/
      jrun_nsapi35.dll" funcs="jruninit,jrunfilter,jrunservice"
  Init proxyport="51000" verbose="false" proxyhost="127.0.0.1"
      timeout="300" rulespath="C:/JRun/servers/default/
      local.properties" fn="jruninit"
  ```

- Adds a JRun object definition. Object definitions are groupings of directives that apply to a particular resource. A typical JRun object definition resembles the following:

  ```
  <Object name="jrun">
  PathCheck fn="jrunfilter"
  Service fn="jrunservice"
  </Object>
  ```

- Adds the following directive in the default object definition:

  ```
  NameTrans fn="jrunfilter"
  ```

  `NameTrans` directives map URLs to physical paths on the host machine. In JRun's case, however, the `NameTrans` directive specifies the `jrunfilter` partial path, so the server will process an object whose `PathCheck` directive matches that partial path (in this case, the `jrun` object).

- Comments out the default URL-to-directory mapping line for `/servlet` so that NES does not override JRun:

  ```
  #NameTrans fn="pfx2dir" from="/servlet"
      dir="C:/Netscape/Server4/docs/servlet" name="ServletByExt"
  ```

Java (non-native) connector

JRun makes the following changes to the `obj.conf` file when connecting to Netscape's Web server using a Java (non-native) connector.

- Prepends `jrun.jar` to the init classpath line.
- Adds a JRun object definition. Object definitions are groupings of directives that apply to a particular resource. A typical JRun object definition with a Java connector resembles the following:

  ```
  <Object name="jrun">
  ```

```
Service fn="java-run" class="com/allaire/jrun/connector/
    JRunConnector" proxyhost="127.0.0.1" proxyport="51000"
    timeout="300"
</Object>
```

- Comments out the default URL-to-directory mapping line for `/servlet` so that NES does not override JRun:

```
#NameTrans fn="pfx2dir" from="/servlet"
    dir="C:/Netscape/Server4/docs/servlet" name="ServletByExt"
```

- Adds the following directives in the default object definition:

```
NameTrans name="jrun" from="*.shtml" fn="assign-name"
NameTrans name="jrun" from="*.jsp" fn="assign-name"
NameTrans name="jrun" from="/servlet/*" fn="assign-name"
```

`NameTrans` directives map URLs to physical paths on the host machine. In JRun's case, however, the `NameTrans` directive specifies the `jrun` object to process the directives.

Sample obj.conf file

The following example provides a sample `obj.conf` file with the modified sections indicated in bold. This file contains changes typical to an NES implementation using the *native* connector.

Sample obj.conf file

```
# Netscape Communications Corporation - obj.conf
# Use only forward slashes in pathnames--backslashes can cause
# problems. See the documentation for more information.

Init fn=flex-init access="C:/Netscape/SuiteSpot/https-tford1/logs/access"
format.access="%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] \"%Req->reqpb.clf-
request%\" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length%"
Init fn=load-types mime-types=mime.types

Init fn="load-modules" shlib="C:/JRun/connectors/nsapi/intel-win/jrun_nsapi35.dll"
funcs="jruninit,jrunfilter,jrunservice"
Init proxyport="51000" verbose="false" proxyhost="127.0.0.1" timeout="300"
rulespath="C:/JRun/jsm-default/services/jse/properties/rules.properties"
fn="jruninit"

<Object name="default">
Nametrans fn="jrunfilter"
NameTrans fn=pfx2dir from=/ns-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir="C:/Netscape/SuiteSpot/manual/https/ug"
NameTrans fn=document-root root="C:/Netscape/SuiteSpot/docs"
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>

<Object name=cgi>
ObjectType fn=force-type type=magnus-internal/cgi
Service fn=send-cgi
</Object>

<Object name="jrun">
PathCheck fn="jrunfilter"
Service fn="jrunservice"
</Object>
```

Configuring WebSite Pro

This section contains configuration information for O'Reilly's WebSite Pro Web server. You must perform all configuration steps described in this section for JRun to communicate with WebSite Pro.

Before beginning, ensure that you have installed WebSite Pro from the WebSite Pro CD and installed the latest WebSite Pro patches from O'Reilly's Web site. Next, install JRun and follow all steps for installing JRun for WebSite Pro.

The following steps are necessary to connect JRun to WebSite Pro:

- "Mapping a URL prefix to run servlets" on page 61
- "Multi-homing and URL prefixes" on page 63
- "Mapping file extensions to JRun" on page 64
- "Configuring JRun to communicate with WebSite Pro" on page 66

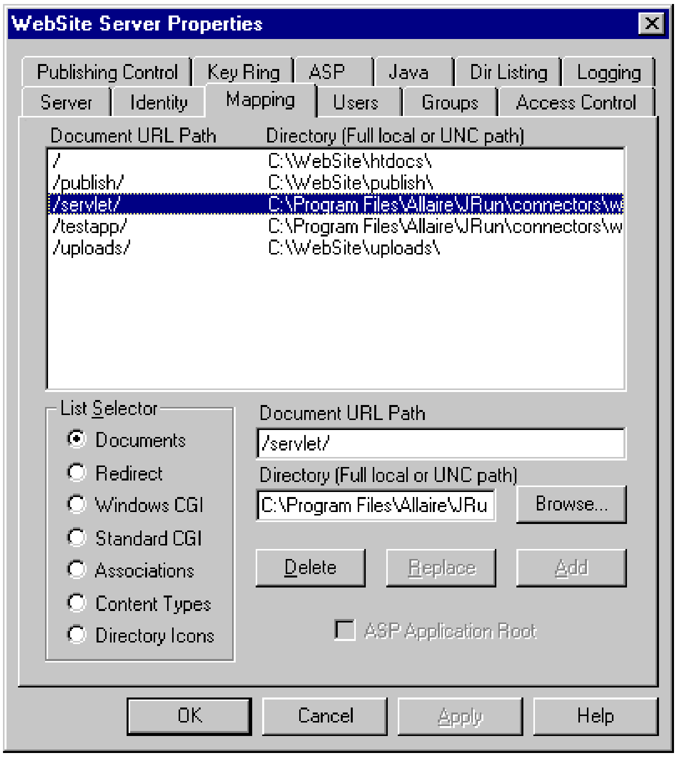
If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you on how to launch the Connector Wizard from within the JMC.

Mapping a URL prefix to run servlets

WebSite Pro allows a great deal of high-level customization, including mapping specific URL prefixes to run servlets. For example, to run a servlet via `http://yourhost.com/servlet/SampleServlet`, you must add a `Documents` mapping for `\servlet\` to WebSite Pro.

To map a URL prefix to JRun:

1. Invoke the WebSite Server Properties application.
2. Select the Mapping tab.



Document URL Path	Directory (Full local or UNC path)
/	C:\WebSite\htdocs\
/publish/	C:\WebSite\publish\
/servlet/	C:\Program Files\Allaire\JRun\connectors\w
/testapp/	C:\Program Files\Allaire\JRun\connectors\w
/uploads/	C:\WebSite\uploads\

List Selector: [● Documents] ○ Redirect ○ Windows CGI ○ Standard CGI ○ Associations ○ Content Types ○ Directory Icons

Document URL Path: /servlet/

Directory (Full local or UNC path): C:\Program Files\Allaire\JRu

3. Edit the `Document URL Path` and `Directory` fields. This enables you to run any servlet via the `\servlet\<`*`servlet_name`*`>` URL.

   For example, your `\servlet\` mapping (in the `Document URL Path` field) might point to `C:\Program Files\Allaire\JRun\connectors\wsapi\intel-win\jrun.isa\servlet\` (in the `Directory` field). You must manually enter the file name at the end of the `Directory` field.

   If you use multiple identities in WebSite Pro, refer to “Multi-homing and URL prefixes” on page 63 for additional information on setting this path.

4. To map a specific servlet to a URL, follow the same steps as described here but add the servlet name to the end. For example:

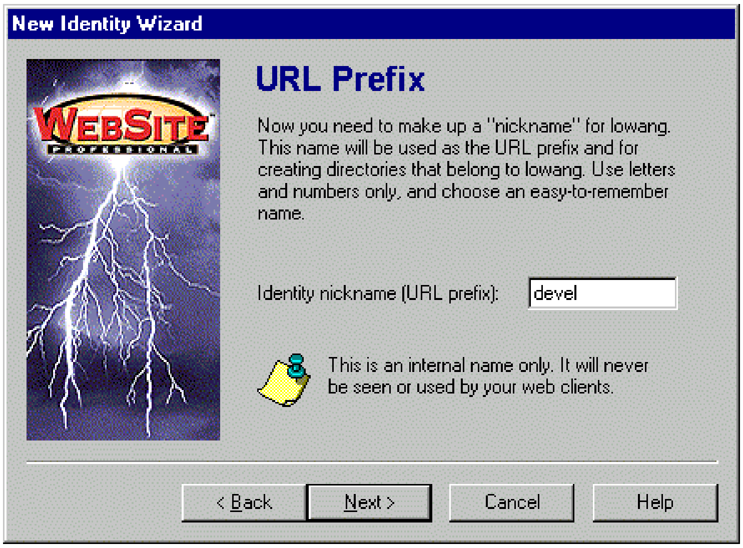
   ```
   C:\Program Files\Allaire\JRun\connectors\wsapi\intel-
   win\jrun.isa\servlet\SnoopServlet
   ```

5. Click OK.

6. Restart your Web server.

Multi-homing and URL prefixes

If you use multiple identities in WebSite Pro, you will need to prepend to your URL mappings the nickname that is assigned to your identity. When setting up your identity, you are asked for a nickname.


New Identity Wizard

URL Prefix

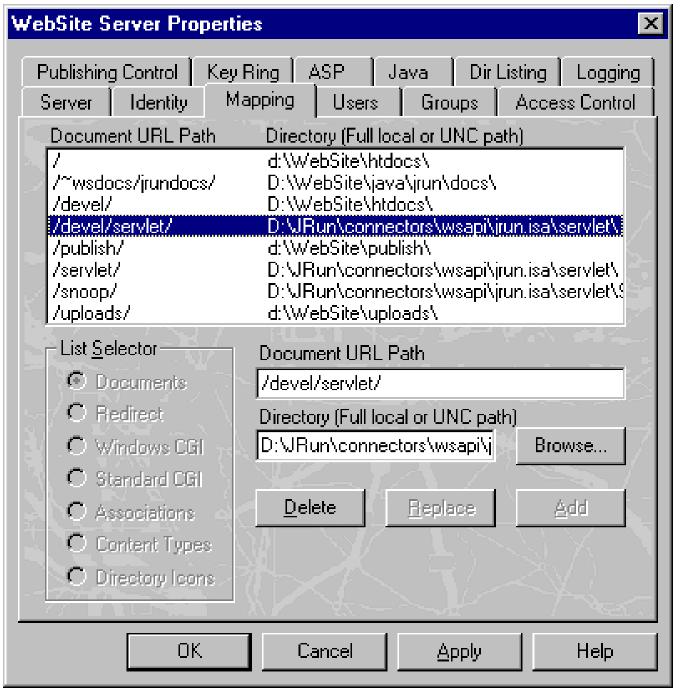
Now you need to make up a "nickname" for lowang. This name will be used as the URL prefix and for creating directories that belong to lowang. Use letters and numbers only, and choose an easy-to-remember name.

Identity nickname (URL prefix): devel

This is an internal name only. It will never be seen or used by your web clients.

This nickname also appears in the `URL Prefix` field in the Identity tab.

This nickname should be prepended to your Document URL Path mappings. For example, to map the `/devel` identity to run servlets, enter `/devel/servlet/` as the Document URL Path.



Document URL Path	Directory (Full local or UNC path)
/	d:\WebSite\htdocs\
/~wsdocs/jrundocs/	D:\WebSite\java\jrun\docs\
/devel/	D:\WebSite\htdocs\
/devel/servlet/	D:\JRun\connectors\wsapi\jrun.isa\servlet\
/publish/	d:\WebSite\publish\
/servlet/	D:\JRun\connectors\wsapi\jrun.isa\servlet\
/snoop/	D:\JRun\connectors\wsapi\jrun.isa\servlet\
/uploads/	d:\WebSite\uploads\

Document URL Path: /devel/servlet/

Directory (Full local or UNC path): D:\JRun\connectors\wsapi\j

Mapping file extensions to JRun

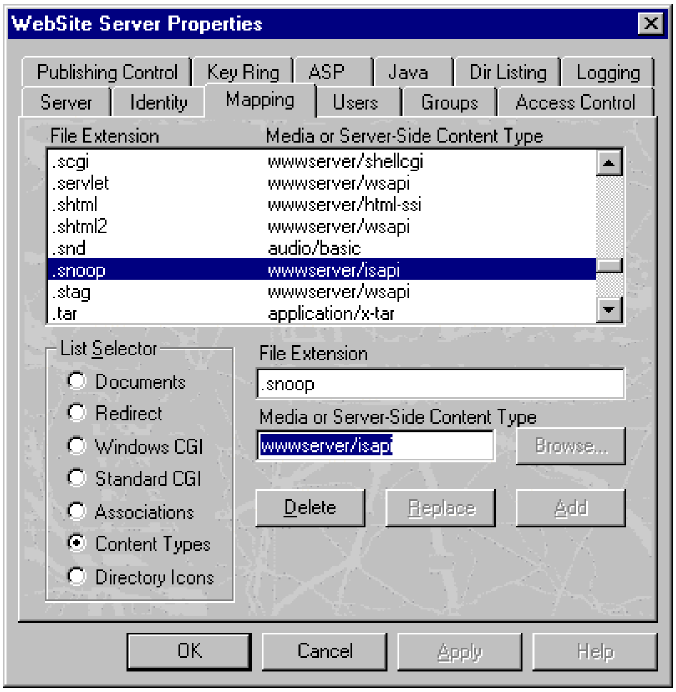
Configuring your WebSite server so that specified file extensions trigger JRun is a two step process. First, you must set up WebSite using it's WebSite Server Properties application. Second, you must use the JRun Management Console to add that mapping to JRun.

The procedure in this section describes how to configure WebSite to map file extensions. For instructions on how to map file extensions in JRun, refer to "Mapping requests to servlets" on page 128.

To add a file extension mapping:

1. Invoke the WebSite Server Properties application.
2. Select the Mapping tab.
3. Select Content Types (in the List Selector box) and add an entry for your extension that maps to `wwwserver/isapi`.

The following figure shows an example for the .snoop extension.


WebSite Server Properties

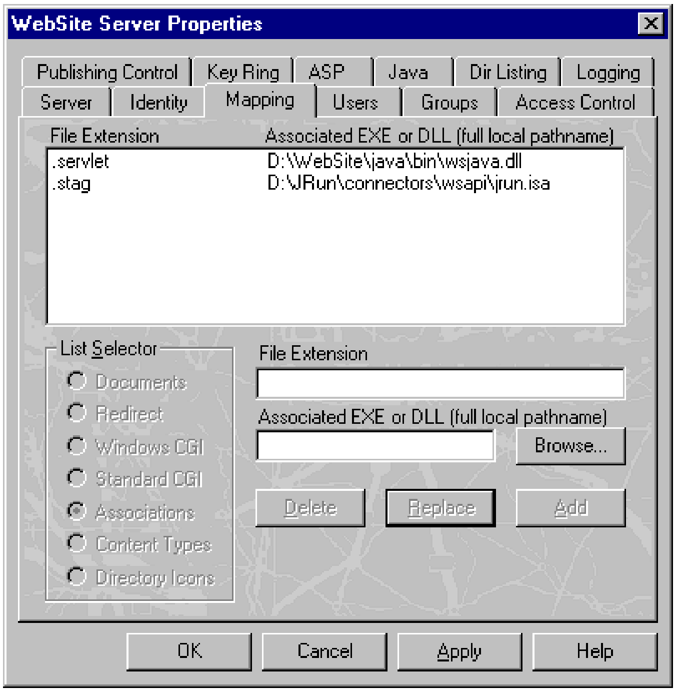
File Extension	Media or Server-Side Content Type
.scgi	wwwserver/shellcgi
.servlet	wwwserver/wsapi
.shtml	wwwserver/html-ssi
.shtml2	wwwserver/wsapi
.snd	audio/basic
.snoop	wwwserver/isapi
.stag	wwwserver/wsapi
.tar	application/x-tar

List Selector: Documents, Redirect, Windows CGI, Standard CGI, Associations, [✓ Content Types], Directory Icons

File Extension: .snoop

Media or Server-Side Content Type: wwwserver/isapi

4. Select Associations (in the List Selector box) and add an entry that maps .snoop to the location of your jrun.isa file, as shown in the following figure.



File Extension	Associated EXE or DLL (full local pathname)
.servlet	D:\WebSite\java\bin\wsjava.dll
.stag	D:\JRun\connectors\wsapi\jrun.isa

List Selector: Associations

5. Click OK.
6. Restart your Web server.

Configuring JRun to communicate with WebSite Pro

This section describes how to configure WebSite Pro to communicate with JRun.

To connect JRun and WebSite Pro:

1. Stop your Web server.
2. Start the JMC using one of the following methods:
   - Select `Start > Programs > JRun 3.0 > JRun Management Console`
   - Open the following URL in your Web browser:

     ```
     http://localhost:8000
     ```

   This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).
3. Log in to the JMC as the JRun administrator.
4. Select `connector wizard` in the access bar.

5. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings		
Connector Wizard Step	**Parameter**	**Description**
Step 1	JRun Server Name	Select the JRun server you want to connect to WebSite Pro. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `WebSite Pro` from the drop-down list.
	Web Server Version	Select WebSite Pro's version from the drop-down list.
	Web Server Platform	Choose the platform on which WebSite Pro is running.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to WebSite Pro. Use the default value of `127.0.0.1`, unless WebSite Pro uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with WebSite Pro. Do not confuse this with WebSite Pro's HTTP port. This book uses `51000` in the examples, but you can select any open port.

6. After you have successfully installed the JRun connector, restart your WebSite Pro Web server and the `default` JRun server.

   If the `default` JRun server is not already running:

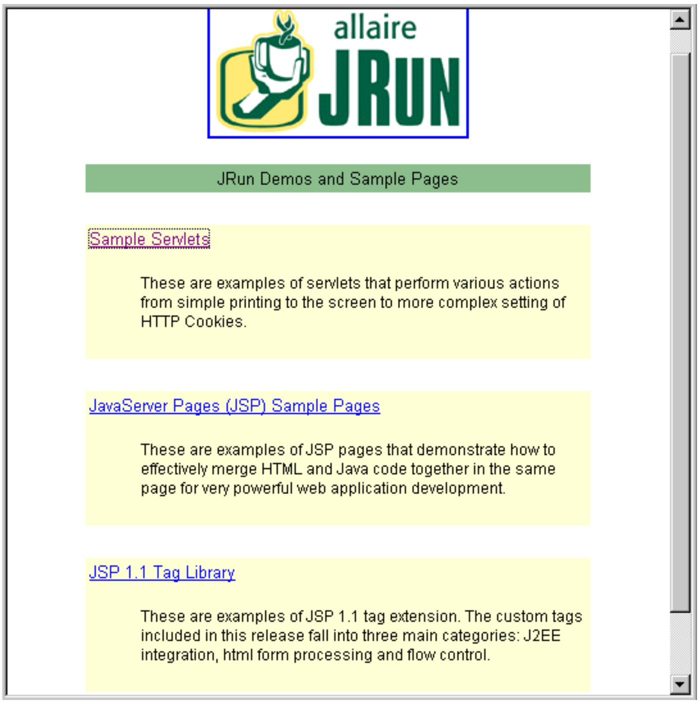
   - If you installed JRun as an application, select `Start > Programs > JRun 3.0 > JRun Default Server`.
   - If you installed JRun as a service, open the Services Control Manager utility (found in the Control Panel) and start the "`JRun Default Server`" service or use the JRun command-line utility:

     ```
     jrun -start default
     ```

7. Verify the JRun connection to your WebSite Pro Web server by running the JRun demo application with the following URL:

```
http://localhost:80/demo/index.html
```

This assumes that WebSite Pro is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

If the demo application runs, you have successfully configured the connection between JRun and your WebSite Pro Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Changes to WebSite Pro configuration files

The JRun Connector Wizard modifies the configuration files of the WebSite Pro Web server. To make any changes to these settings not detailed in this document, use the WebSite Pro Server Properties interface.

The Connector Wizard also changes the JRun `local.properties` files. For more information, refer to “Changes to local.properties” on page 72.

Configuring Java-Based Web Servers

JRun can communicate with most Java Web Servers, whether or not the procedures are detailed in this document for each particular server. This section describes how to configure Java-based Web servers that are not covered in detail in other sections of this chapter.

To connect JRun and Java-based Web servers:

1. Copy the `jrun.jar` file from the JRun distribution into the `/lib/classes` directory of your Web server. (Some applications may require you to add `jrun.jar` to the classpath; see the documentation provided with your Web server.)
2. Create an alias named `JRunConnector` that points to the following class:

   ```
   allaire.jrun.connector.JRunConnector
   ```
3. Set up two initialization parameters as follows:

   ```
   proxyhost=localhost
   proxyport=51000
   ```

   The above values are defaults and do not have to be set explicitly unless you want to use different values. If you have configured JRun differently, you may need to change the port number.
4. Use your Web server's administrative interface to map the appropriate HTTP requests to JRun. For example:

   ```
   /servlet=JRunConnector
   *.jsp=JRunConnector
   /msservlets=JRunConnector
   ```

   All requests that match these mappings will be forwarded to JRun.

Configuring a CGI Interface for Running Servlets

JRun supports a Perl connector for Web servers that use CGI. To configure your Web server to use the Perl connector, use the `jrun.pl` Perl script supplied with JRun. This script is located in `<JRun_directory>/connectors/perl5/`.

Copy `jrun.pl` to any directory that you use to run CGI scripts. Assuming that your CGI directory is `cgi-bin`, you can invoke servlets as follows:

```
http://host/cgi-bin/jrun.pl/servlet/SnoopServlet
```

In this example, `/servlet/SnoopServlet` is passed to JRun, and JRun invokes the servlet. The output of the servlet is returned by the CGI script.

Shown below is another example

```
http://host/cgi-bin/jrun.pl/yourpage.jsp
```

This example invokes `/yourpage.jsp` on JRun and return the results.

The `jrun.pl` script looks for the `JRUNPROXY` environment variable to determine how to connect to the JRun server. The value must be in the form:

```
<IP address>:<port>
```

The default value is: `127.0.0.1:51000`

You can customize this script for your specific environment. The JRun proxy address can be changed, and the error responses can be edited by modifying the appropriate subroutine of the script.

Configuring the Zeus Web Server

If you are using a Zeus Web server, perform the steps described below in order to configure your Web server to connect to JRun.

If you are running the Connector Wizard as part of your JRun installation, you can skip the steps that instruct you to launch the Connector Wizard from within the JMC.

To connect JRun and Zeus:

1. Log in to Zeus using the administrative login.
2. Select Module config.
3. Enable the Distributed option.
4. Select the Distributed link.
5. Under the Java Servlets option, enter the following settings:

   ```
   Servlet prefs: /servlet
   Servlet Server:ip_address:port_number
   ```

   Set *`ip_address`* to the IP address of the Web server's host. If the Web server and JRun are on the same host machine, set *`ip_address`* to `127.0.0.1`.

   Set *`port_number`* to the port on which Zeus and JRun communicate. The default is `51000`.
6. Start the JMC using one of the following methods:
   - Select `Start > Programs > JRun 3.0 > JRun Management Console.`
   - Open the following URL in your Web browser:

     `http://localhost:8000`

   This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).
7. Log in to the JMC as the JRun administrator.
8. Select `connector wizard` in the access bar.

9. Specify the necessary configuration information in the Connector Wizard, as described in the following table.

JRun Connection Module Settings

Connector Wizard Step	Parameter	Description
Step 1	JRun Server Name	Select the JRun server you want to connect to the Web server. In most cases, you should select the Default Server. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications.
	Web Server Type	Select `Zeus Web Server` from the drop-down list.
	Web Server Version	Select your Web server's version from the drop-down list.
Step 2	JRun Server IP Address	Enter the IP address of the JRun server that will connect to Zeus. Use the default value of `127.0.0.1`, unless Zeus uses a different IP address than the JRun server.
	JRun Server Connector Port	Enter the port that the JRun server will use to communicate with Zeus. Do not confuse this with Zeus' HTTP port. This book uses `51000` in the examples, but you can select any open port.

10. After you have successfully installed the JRun connector, restart your Zeus Web server and the `default` JRun server.

    If the `default` JRun server is not already running:

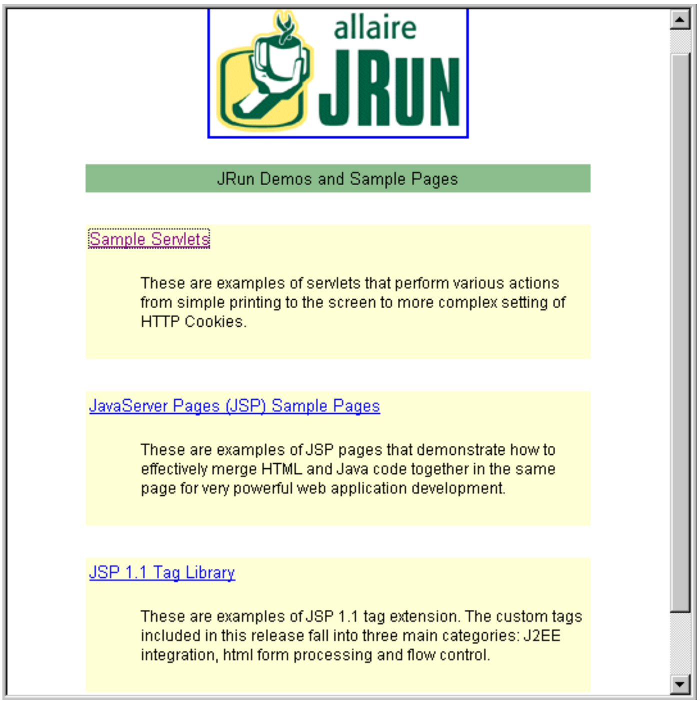
    - If you installed JRun as an application, select `Start > Programs > JRun 3.0 > JRun Default Server`.
    - If you installed JRun as a service, open the Services Control Manager utility (found in the Control Panel) and start the "`JRun Default Server`" service or use the JRun command-line utility:

      ```
      jrun -start default
      ```

11. Verify the JRun connection to your Zeus Web server by running the JRun demo application with the following URL:

    ```
    http://localhost:80/demo/index.html
    ```

This assumes that Zeus is listening for connections on the default port 80.



JRun Demos and Sample Pages

Sample Servlets

These are examples of servlets that perform various actions from simple printing to the screen to more complex setting of HTTP Cookies.

JavaServer Pages (JSP) Sample Pages

These are examples of JSP pages that demonstrate how to effectively merge HTML and Java code together in the same page for very powerful web application development.

JSP 1.1 Tag Library

These are examples of JSP 1.1 tag extension. The custom tags included in this release fall into three main categories: J2EE integration, html form processing and flow control.

If the demo application runs, you have successfully configured the connection between JRun and your Zeus Web server. If the demo application does not run correctly, refer to “Troubleshooting” on page 73.

Changes to Zeus configuration files

The JRun Connector Wizard modifies the configuration files of the Zeus Web server. To make any changes to these settings not detailed in this document, use the Zeus administration interface.

The Connector Wizard also updates the JRun `local.properties` files. For more information, refer to “Changes to local.properties” on page 72.

Changes to local.properties

In addition to Web server configuration files, the JRun Connector Wizard makes the following changes to the JRun server’s `local.properties` file:

- Adds the following line to the end of `local.properties` indicating that the JRun Connector has successfully been installed.

```
ranConnector=yes
```

- Sets the `jcp.endpoint.main.port` in the `jcpservices` section to the port you specified while running the Connector Wizard. This is the `JRun Server Connector Port`, also known as the `Proxy Port`.

Troubleshooting

The information in this section is designed to help you get past the most common problems associated with connecting JRun to an external Web server. For information on troubleshooting the JRun installation, refer to “Troubleshooting” on page 35.

When troubleshooting JRun, you can also check the log files located in `<JRun_directory>/logs` for additional information.

Using the JRun Connector Wizard

The JRun Connector Wizard hides much of the complexity of configuring a connection between you and your Web server. This section describes some of the common errors that might occur while running the Wizard.

If you get one of the following error(s):

```
"httpd.conf not accessible"
"Could not load obj.confile"
"Error copying JRun ISAPI filter"
```

Try one of the following:

- You may have entered a bad path (or no path) to the Web server’s configuration files in step 3 of the Connector Wizard. Rerun the Connector Wizard and enter the correct path.
- Stop your Web server and rerun the Connector Wizard.

Testing the JRun demo application

After installing a connection between your Web server and JRun with the Connector Wizard, verify the connection by running the JRun demo application with the following URL:

```
http://localhost:80/demo/index.html
```

This section describes some of the common errors that might occur while testing the demo application.

HTTP errors

```
"404 File Not Found" error
"The page cannot be found"
"500 Internal Server Error"
```

`"Could not connect to JRun server"`

Try one of the following:

- Make sure the JRun `default` server is running. The demo application is running on this server by default.
  - In Windows, check the system tray. The JRun Server 3.0 icon should appear if you installed JRun as an application. If you installed JRun as a service, check the Services Control Manager utility (found in the Control Panel) to see if the `JRun Default Server` service is running.
  - On UNIX, use the command-line tool in `/opt/bin` to determine if the server is running:

    ```
    jrun -status default
    ```
- If the `default` JRun server was already running, restart it after running the Connector Wizard.
- Check the port number in the request URL. The default HTTP port is 80, but if your Web server is listening on a different port, you must specify this in the URL. For example, if your Web server is listening on 8080, to access the demo application, you would enter `http://localhost:8080/demo/index.html`.
- Make sure your Web server is running after using the JRun Connector Wizard.
- Make sure you have read access for the `/JRun/servers/default` directory and its subdirectories.

Concurrency errors

If you get the following error:

`"Too Many Concurrent requests"`

"*`Reverting to Developer Edition`*" in the JMC

Try one of the following:

- Upgrade your license. The free download and Developer Editions of JRun only allows up to 3 concurrent connections. For more information, refer to “JRun Product Variations” on page 2.
- If you are using a Beta or evaluation version of JRun, you may get this message based on when the evaluation period expires. If this is the case, upgrade to the most recent evaluation or production version.

Process errors

If you get locked out of your system when scripting the jrun command:

Use the `-nohup` option (UNIX only) to spawn a new process for the server in the background rather than creating a child process in the foreground. It acts the same way as the `&`. For example:

```
jrun -nohup -start default
```

If you do not use the `-nohup` option and get locked out of the foreground, press Ctrl + z and then enter `bg` at the prompt to put the JRun server process in the background.

For `-start` option syntax, refer to “Using the JRun command” on page 86.

CHAPTER 3

JRun Management Console

The JRun Management Console (JMC) is a browser-based utility that enables you to configure various settings in JRun. This chapter provides an overview of the JMC and explains the functions that you can perform with it.

Contents

Starting the JRun Management Console

JRun includes the JRun Management Console (JMC), a browser-based Web application used to configure the JRun environment and the connection between JRun and your Web server(s). By default, the JMC runs on the `admin` JRun server.

This section explains how to start the JMC and describes the basic layout and features of the console.

Note Since the JMC is JSP-based, you must use either Netscape Communicator Version 4.0 or later or Internet Explorer Version 4.0 or later to access it.

To start the JMC:

1. Start the `admin` JRun server if it is not already running. For information on starting JRun servers, refer to "Starting and Stopping JRun Servers" on page 33.
2. Start the JMC by opening the following URL in your Web browser:

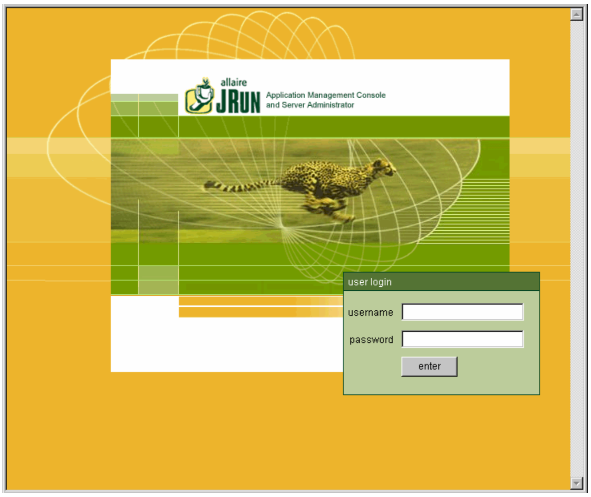
   ```
   http://localhost:8000
   ```

   Or (Windows only): Select `Start > Programs > JRun 3.0 > JRun Management Console`

   Note This procedure assumes you are connecting to the JMC using the JRun-supplied Web server on the default port (8000).

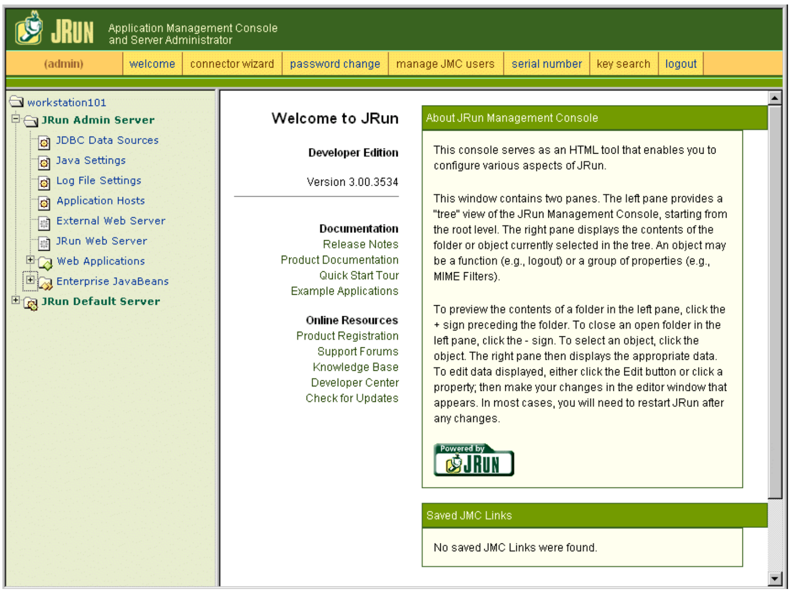
   If this is the first time you are launching the JMC, the JRun license agreement appears. If the JMC does not open, refer to "Troubleshooting" on page 35.
3. Accept the JRun license agreement. (You only have to accept the license agreement once.)

   The JMC login window appears:



4. Enter your username and password in the fields provided and click Login. The default user name is `admin`. You created the password for `admin` during the installation procedure.

   The JMC window appears with the Welcome page open.



Welcome to JRun

Developer Edition

Version 3.00.3534

Documentation
Release Notes
Product Documentation
Quick Start Tour
Example Applications

Online Resources
Product Registration
Support Forums
Knowledge Base
Developer Center
Check for Updates

About JRun Management Console

This console serves as an HTML tool that enables you to configure various aspects of JRun.

This window contains two panes. The left pane provides a "tree" view of the JRun Management Console, starting from the root level. The right pane displays the contents of the folder or object currently selected in the tree. An object may be a function (e.g., logout) or a group of properties (e.g., MIME Filters).

To preview the contents of a folder in the left pane, click the + sign preceding the folder. To close an open folder in the left pane, click the - sign. To select an object, click the object. The right pane then displays the appropriate data. To edit data displayed, either click the Edit button or click a property; then make your changes in the editor window that appears. In most cases, you will need to restart JRun after any changes.

Saved JMC Links

No saved JMC Links were found.

This window displays two panes. The left pane provides a tree view of the JRun object hierarchy, starting at the machine level. The right pane displays the contents of the folder or object currently selected in the tree. Above the panes is the access bar. This bar contains commands that are not JRun-server specific.

If you do not have `admin` privileges, you will not see all the options in the access bar or be able to access all the objects in the tree.

An object in the tree may be a function (for example, Change Password) or a property (for example, `serial_number`).

Using the JRun Management Console

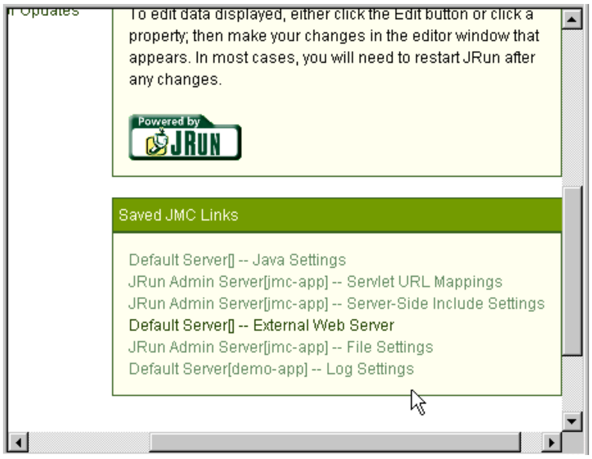
When using the console, note the following:

- To preview the contents of a folder in the left pane, click the + sign preceding the folder.
- To close an open folder in the left pane, click the - sign.
- To display the contents of a folder in the right pane, click the folder. In this document, the greater than sign (>) indicates levels of folders, subfolders, and files or objects, and italics indicates variable information. For example, "`Select` *`machine_name`* `>` *`JRun_server_name`* `>` *`application_name`* `> Logging`."

- To select an object, click the object. The right pane then displays the properties of that object or JRun executes the function. To edit properties, either click the Edit button in the right pane or click a property; then make your changes in the editor window that appears.
- In most cases, you must restart JRun after making any changes to the JRun server properties in order for your changes to take effect.

Setting JMC favorites

The JMC allows you to add a list of commonly used edit windows on the Welcome page. For example, if you make changes to the session settings of a particular Web application often, you can add a link so that you can get to the page in one click from the Welcome page rather than traversing the object explorer.



To edit data displayed, either click the Edit button or click a property; then make your changes in the editor window that appears. In most cases, you will need to restart JRun after any changes.

Saved JMC Links

Default Server[] -- Java Settings
JRun Admin Server[jmc-app] -- Servlet URL Mappings
JRun Admin Server[jmc-app] -- Server-Side Include Settings
Default Server[] -- External Web Server
JRun Admin Server[jmc-app] -- File Settings
Default Server[demo-app] -- Log Settings

To add a link to a specific panel in the JMC, click the "Add to Welcome Page" checkbox in the lower right corner of the panel. JRun confirms that the panel was added. The next time you go to the Welcome page, the link appears in the Saved JMC Links section.

Setting the JRun Serial Number

The JRun serial number defines the version of JRun you are running. If you installed JRun Developer edition or the free evaluation copy, the serial number is blank. If you upgrade your JRun license, you can change the serial number in the JMC to unlock the new functionality.

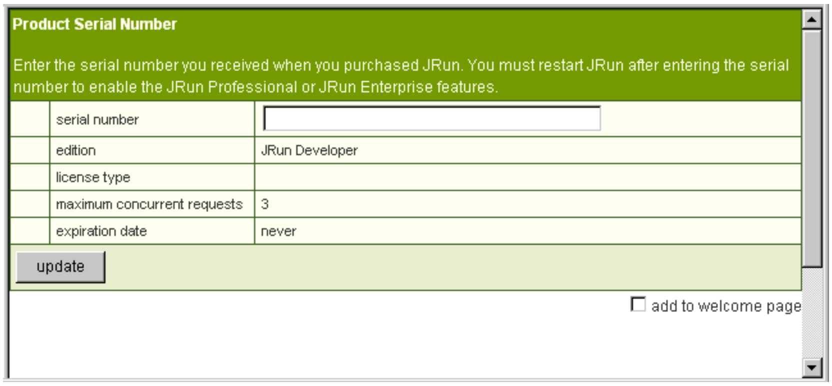
This section describes how to change your JRun serial number. For more information on purchasing a JRun license, refer to “Contacting Allaire” on page 13.

Note Only a user logged in as `admin` can change the `Serial Number` property.

To modify your serial number:

1. Select `Serial Number` in the access bar. The access bar is located across the top of the JMC panes.

   The Product Serial Number panel appears.



Product Serial Number

Enter the serial number you received when you purchased JRun. You must restart JRun after entering the serial number to enable the JRun Professional or JRun Enterprise features.

	serial number	
	edition	JRun Developer
	license type	
	maximum concurrent requests	3
	expiration date	never

2. In the `Serial Number` field, enter the serial number, *exactly* as it was provided to you by Allaire Corporation.
3. To apply your changes, click `Update`.
4. Restart your JRun servers.

Managing JMC Users

The JMC provides utilities for managing users of the JMC. This allows the JRun administrator to restrict access by JRun server. For example, if you operate an ISP, you can give each customer their own JRun server with access rights to the settings of that server only. Users cannot see JRun servers in the JMC that they do not have permission to access. This section describes how to add and remove users and change user settings.

If you make changes to the current user, the changes will not take effect until you log out and log back in. If you make changes to another user, the changes will not take effect until you log out and that user logs in.

To identify what user you are currently logged in as, look in the left side of the JMC's access bar. The first time you log in, this should appear as `(admin)`.

Adding new JMC users

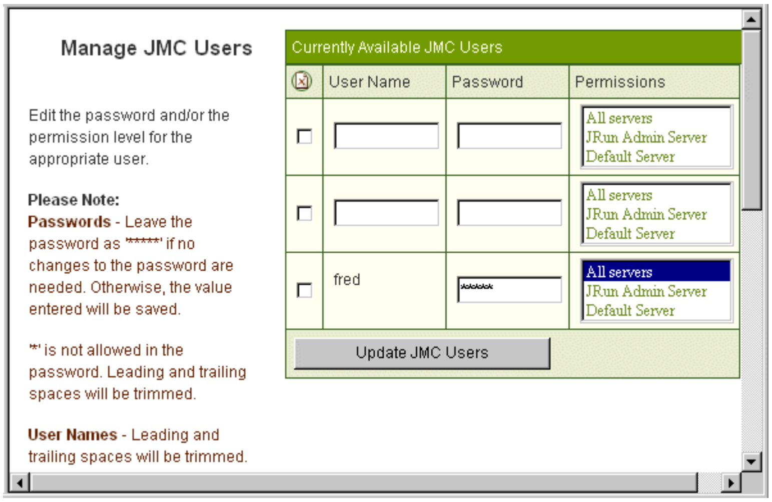
Using the Manage JMC Users option, you can add users with different levels of access to the JMC. You can give each user access to some, all, or none of the JRun servers.

Note Only a user logged in as `admin` can access the Manage JMC Users option.

To add a new user:

1. Select `Manage JMC Users` in the access bar.

   The JMC User Manager panel appears in the right pane.



Manage JMC Users

Edit the password and/or the permission level for the appropriate user.

Please Note:
Passwords - Leave the password as '*****' if no changes to the password are needed. Otherwise, the value entered will be saved.

'*' is not allowed in the password. Leading and trailing spaces will be trimmed.

User Names - Leading and trailing spaces will be trimmed.

Currently Available JMC Users

	User Name	Password	Permissions
☐			All servers, JRun Admin Server, Default Server
☐			All servers, JRun Admin Server, Default Server
☐	fred	*****	All servers (selected), JRun Admin Server, Default Server

2. Enter a name for the new user in the `User Name` field.

   Note You can use spaces in the `User Name`. For example, "`Nick Danger`" is a valid name. Leading and trailing spaces will be trimmed.

3. Enter a password for the new user in the `Password` field. When entering a new user's password, keep the following in mind:
   - The password is not masked by asterisks while entering it in the JMC User Manager panel.
   - The minimum password length is one character.
   - You cannot use spaces or the asterisk character (`*`) in the password.

4. Select the JRun server(s) you want the new user to have access to in the `Permissions` listbox. To select multiple JRun servers, click on the first one and hold the `Ctrl` key down while selecting additional JRun servers.

5. To apply your changes, click Update JMC Users.

Changing JMC user settings

Using the Manage JMC Users option, you can change user passwords or change which JRun servers users have access to. You cannot change the settings for the `admin` user using the Manage JMC Users option.

Note Only a user logged in as `admin` can access the Manage JMC Users option.

To change user settings:

1. Select `Manage JMC Users` in the access bar.

   The JMC User Manager appears in the right pane.
2. To change JRun server access for a user, select the JRun servers in the `Permissions` listbox for that user. To select multiple JRun servers, click on the first one and hold the `Ctrl` key down while selecting additional JRun servers.
3. To change a user's password, select the masked password in the `Password` field and enter a new password. When entering a new password, keep the following in mind:
   - The new password is not masked by asterisks while entering it in the JMC User Manager.
   - The minimum password length is one character.
   - You cannot use spaces or the asterisk character (*) in the password.
4. To apply your changes, click Update JMC Users.

These changes will take effect the next time that user logs in.

Removing JMC users

Using the Manage JMC Users option, you can remove any user from JRun except the `admin`. Only a user logged in as the `admin` can access the Manage JMC Users option.

Note Make sure users are logged out of the JMC before you remove them.

To remove a user:

1. Select `Manage JMC Users` in the access bar.

   The JMC User Manager panel appears in the right pane.
2. Select the `Delete` checkbox ☒ for the user you want to remove. You can select any number of users to delete at one time.
3. To apply your changes, click Update JMC Users.

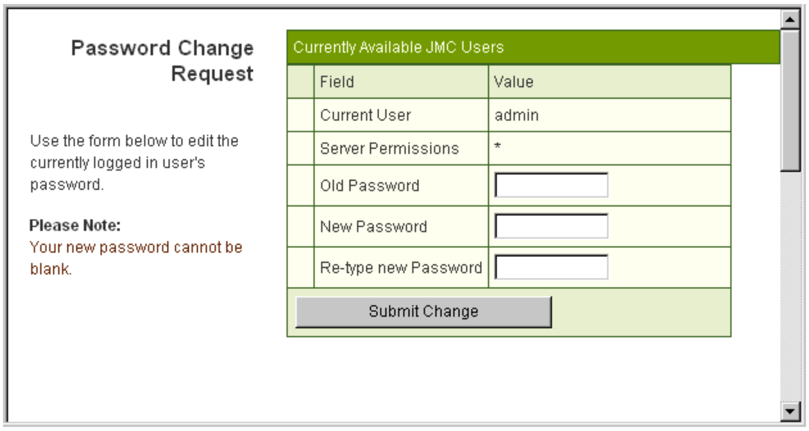
Changing your password

Using the Password Change option in the JMC, you can change the password of the currently logged-in user. If you are logged in as `admin`, you can also change any other user's password using the Manage JMC Users option. For more information, refer to "Changing JMC user settings" on page 82.

To change your password:

1. Select `Password Change` in the access bar.

   The Password Change Request panel appears in the right pane, showing the currently logged-in user and JRun server permissions.



Password Change Request

Use the form below to edit the currently logged in user's password.

Please Note:
Your new password cannot be blank.

Currently Available JMC Users	
Field	Value
Current User	admin
Server Permissions	*
Old Password	
New Password	
Re-type new Password	

   An asterisk (`*`) in the `Server Permissions` field indicates that the user has universal access.

2. Enter your old password in the `Old Password` field.
3. Enter your new password in `New Password` field the and confirm it in the `Re-type New Password` field. When entering a new password, consider the following:
   - The password is masked by asterisks while entering it in the `Password Change Request` window.
   - You cannot use spaces or the asterisk (`*`) character in the password.
   - The minimum password length is one character.
4. To apply your changes, click Submit Change.

Configuring JRun Servers

JRun servers form the core of the JRun architecture. They perform the following functions:

- Provide a logical way of grouping Web applications. Any number of Web applications can run within a JRun server.
- Connect Web applications to both internal and external Web servers (via JRun Connection Modules).

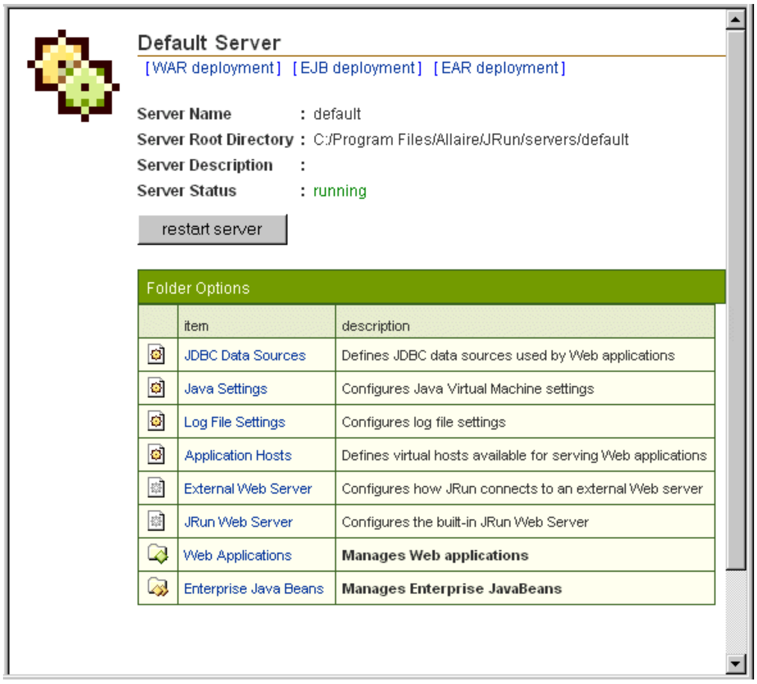
- Maintain Web server stability. Each JRun server runs as its own process. The services provided by the JRun server are also out-of-process.
- Allow you to implement your business logic through Enterprise JavaBeans.

The JRun installation sets up two JRun servers: `default` and `admin`. The `default` server appears in the JMC as `Default Server` and the `admin` server appears as `JRun Admin Server`.

The `admin` server provides you with access to the `JRun Management Console` application (`jmc-app`). The `default` server is provided to get you up and running quickly and includes an empty `Default User Application` (`default-app`) and the `JRun Demo` (`demo-app`).

Note On Windows NT, JRun servers can be run as either NT services or applications. As services, the JRun Admin Server and JRun Default Server start up every time the machine boots and run as system processes rather than as user processes. As applications, the JRun servers must be started manually.

The JMC provides the JRun Server panel that you can access by selecting *`machine_name > JRun_server_name`*.



Server Name : default
Server Root Directory : C:/Program Files/Allaire/JRun/servers/default
Server Description :
Server Status : running

	item	description
	JDBC Data Sources	Defines JDBC data sources used by Web applications
	Java Settings	Configures Java Virtual Machine settings
	Log File Settings	Configures log file settings
	Application Hosts	Defines virtual hosts available for serving Web applications
	External Web Server	Configures how JRun connects to an external Web server
	JRun Web Server	Configures the built-in JRun Web Server
	Web Applications	**Manages Web applications**
	Enterprise Java Beans	**Manages Enterprise JavaBeans**

The JRun Server panel shows you information about the server as well as the status of the server. You can also use this panel to restart the server, as described in "Restarting JRun Servers in the JMC" on page 86.

This section describes the following procedures using the JMC:

- "Restarting JRun Servers in the JMC" on page 86
- "Using the JRun command" on page 86
- "Adding and removing JRun servers" on page 88
- "Configuring Java Virtual Machines" on page 91
- "Configuring JRun server event logs" on page 94

For additional information on working with JRun servers, refer to *Developing Applications with JRun*.

Restarting JRun Servers in the JMC

The JMC provides you with an easy way to restart the JRun servers for most of the changes made in the JMC to take effect.

In UNIX and Windows, you can also restart the JRun servers from the command-line. For more information, refer to "Using the JRun command" on page 86.

In Windows, you can also restart JRun servers from the Services Control Manager utility (found in the Control Panel) if you run the JRun servers as services or from the system tray (if you run them as applications).

To restart a JRun server in the JMC:

1. Select *`machine_name`* `>` *`JRun_server_name`*.

   The JRun Server panel appears.
2. Click Restart Server.

   Note Do not restart the `admin` server from the JMC since the JMC is running, by default, on the that server.

Using the JRun command

JRun provides a command-line utility for use in both the Windows and UNIX environments. This section explains the options for this utility.

On Windows, run the command-line utility from `<`*`JRun_directory`*`>/bin`. On UNIX, it is located in the `/opt/jrun/bin` directory. To list the `jrun` command options, enter "`jrun`" (without options) at the command-line.

The syntax for this command is:

```
jrun -[admin | console | demo | install | java | remove | restart |
[-nohup] start | status | stop ] [parameters]
```

admin

```
jrun -admin
```

Starts the JRun Management Console on Windows (NT only). This command takes no parameters.

console

```
jrun -console
```

UNIX only. Disables the stdout/sterr redirection specified by `java.System.out` and `java.System.err` in the `local.properties` file so that these are output to the console rather than a log file.

demo

```
jrun -demo default
```

Starts the JRun Demo application on the `default` server. If you redeploy the `demo` application to another JRun server, specify that server in the command-line. Windows NT only.

install

```
jrun -install NT-service_name server_name -[quiet]
```

Installs JRun as a Windows NT service (NT only). *`server_name`* must be one of the JRun servers from the `jvms.property` file. Add the real path of the server's root directory to the `jvms.properties` file. For example: `"C:\Program Files\Allaire\JRun\servers\foo"`.

For example:

```
jrun -install "Foo Service" foo -quiet
```

The `-quiet` option prevents a dialog box from appearing regardless of whether the command is successful or not.

java

```
jrun -java java-prog -classpath path [java-args] class [class-args]
```

Launches a Java application other than JRun. Use the `-classpath` option to specify a directory. JRun includes all the `.jar` files in that directory.

For example:

```
jrun -java c:\jdk1.2.2\bin\java -classpath c:\JRun\lib JRun -start _
c:\JRun\servers\default
```

remove

```
jrun -remove NT-service-name -[quiet]
```

Removes a JRun server that is a Windows NT service (NT only).

For example:

```
jrun -remove "JRun Default" -quiet
```

The `-quiet` option prevents a dialog box from appearing regardless of whether the command is successful or not.

restart

```
jrun -restart [JRun_server_name]
```

Restarts all JRun servers. Specify `JRun_server_name` to restart only that JRun server.

start

```
jrun [-nohup] -start [JRun_server_name]
```

Starts all JRun servers. Specify `JRun_server_name` to start only that JRun server.

The `-nohup` option (UNIX only) spawns a new process for the server in the background rather than creating a child process in the foreground.

status

```
jrun -status [JRun_server_name]
```

Displays the status of all JRun servers. Specify `JRun_server_name` to display the status of only that JRun server.

stop

```
jrun -stop [JRun_server_name]
```

Stops all JRun servers. Specify `JRun_server_name` to stop only that JRun server.

Adding and removing JRun servers

The default installation of JRun includes two servers, `admin` and `default`. These JRun servers contain sample applications and provide a way to get you up and running quickly. You may want to add additional JRun servers to keep the QA, production, and development environments separate, or if you develop applications for deployment on multiple Web sites.

Adding a JRun server

The easiest way to create a new JRun server is to copy the `default` JRun server's files and directory structure, including the `default` application, and then modify the copy to reflect your new server.

To add a JRun server:

1. Copy the `/servers/default` directory to a directory with the new server name (for example, `/servers/foo`).
2. Open the `local.properties` file in the new directory (`/servers/foo`). The `local.properties` file defines the server-specific settings of a JRun server, including basic properties such as name and port settings, as well as what applications are deployed in the server.

   Make the following changes to the new `local.properties` file:

a. Change the `jrun.server.displayname` property to reflect the new name of the server. The code should now look something like this:

```
## jvm properties
# was: jrun.server.displayname=Default Server
jrun.server.displayname=Foo Server
```

b. Remove applications other than the `default-app` in the `Web Application Settings` section. If you have not added any applications to the `default` server, you do not need to delete anything other than the `demo-app`.

For example, each application might have a section that looks like this:

```
jmc-app.rootdir=C:\\ProgramFiles\\Allaire\\JRun\\servers _
    \\admin\\jmc-app
jmc-app.class={webapp.service-class}
webapp.mapping./=jmc-app
```

Delete that application's settings unless it refers to the `default-app`.

c. Reset the `servlet.webapps` property to contain only the default application. The line should now look like this:

```
#was: servlet.webapps=default-app, demo-app
servlet.webapps=
```

d. Change the port settings (these settings are located in different sections of the file). For example:

```
web.endpoint.main.port=8101 #was:8100 (Web server port)
jcp.endpoint.main.port=51001 #was:51000 (Listening port)
control.endpoint.main.port=53001 #was: 53000 (Control port)
ejipt.classServer.port=2324 #was: 2323
ejipt.homePort=2334 #was 2333
```

Note If you do not change these port settings, you may experience binding exceptions.

3. Save the `local.properties` file.
4. Delete the directories of all applications other than the `default-app` from `/servers/foo`. If you have not added any applications to the `default` server, then you will not have to delete any applications.
5. Open the `jvms.properties` file in the *<JRun_rootdirectory/*`lib` directory. JRun uses the `jvms.properties` file to determine what servers should be instantiated. Add a line for the new server. The file should appear as follows (on NT):

```
admin=C:/Program Files/Allaire/JRun/servers/admin
default=C:/Program Files/Allaire/JRun/servers/default
foo=C:/Program Files/Allaire/JRun/servers/foo
```

Take note of the server name at the beginning, as well as the directory name at the end of the line.

6. Start the new JRun server:

Windows:

a. In <*JRun_rootdirectory*>/bin, copy `jrun-default.bat` file to `jrun-foo.bat`.

b. Edit `jrun-foo.bat` to specify the name of the new server:

```
@echo off
start jrun -start foo
```

c. Run the `jrun-foo.bat` file.

d. Optionally (NT only), you can also add the new server as an NT service using the command-line utility:

```
jrun -install "Foo Service" foo -quiet
```

e. Start the server using the following command:

```
jrun -start foo
```

UNIX:

Use the `jrun` command-line utility in `/jrun/bin`:

```
jrun -start foo
```

7. You can give the new JRun server a more descriptive mnemonic in the JMC. In the `jvm properties` section of the `local.properties` file, set the `jrun.server.displayname` property to a "pretty name" you want your server to appear as in the JMC. For example:

```
jrun.server.displayname=Foo Server
```

8. If you are logged in to the JMC, log out and restart the `admin` server.

The next time you log in, the new `Foo Server` appears in the left pane of the JMC. It should have no Web applications in it.

Removing a JRun server

Removing a JRun server should be done with great consideration. Before removing a server, be sure to back up any important files or applications residing within that server. This section describes how to remove a JRun server.

Caution Do not remove the `admin` or `default` JRun servers. Applications in these servers will be lost.

To remove a JRun server:

1. Stop the JRun server you want to remove.
2. Delete the server's directory and all subdirectories (for example, delete the `/foo` directory in `/servers`).
3. Open the `jvms.properties` file in the <*JRun_rootdirectory*/>`lib` directory and remove the server's line. The file should look like this (on NT):

```
admin=C:/Program Files/Allaire/JRun/servers/admin
default=C:/Program Files/Allaire/JRun/servers/default
##foo=C:/Program Files/Allaire/JRun/servers/foo
```

4. Remove any startup scripts for the server.

5. If you added the server as an NT service, use the command-line utility to remove this service:

   ```
   jrun -remove "Foo Server" -quiet
   ```

6. If you are logged in to the JMC, log out and restart the `admin` server.

   The next time you log in, the server does not appear in the left pane of the JMC.

Configuring Java Virtual Machines

The Java Virtual Machine (JVM), also known as a JRE, is the software implementation of a CPU. It contains everything necessary to run programs written for the Java platform.

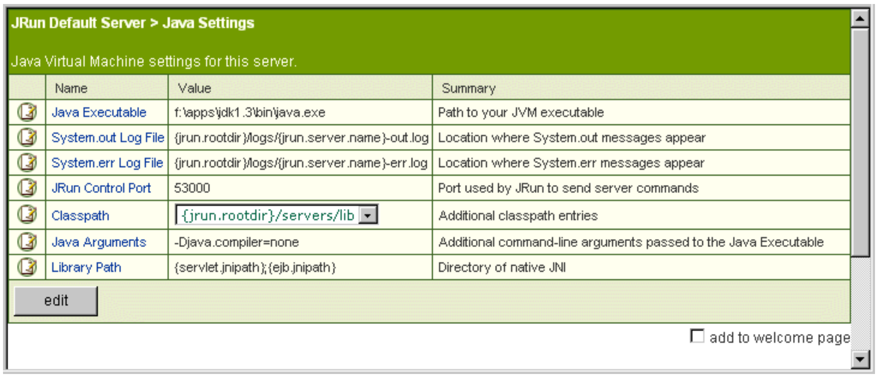
Each JRun server has associated with it a single JVM that executes all servlets, JSP pages, and EJBs for that JRun server. Use the information in this section to configure the JVM for each JRun server.

The Java Settings panel allows you to set the log file output locations. On UNIX systems, you can also redirect the `Java.System.err` and `java.System.out` output to the console using the `-console` option of the `jrun` command. For more information, refer to “Using the JRun command” on page 86.

To edit general settings for a JVM:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Java Settings`.

   The Java Settings panel appears.



JRun Default Server > Java Settings

Java Virtual Machine settings for this server.

	Name	Value	Summary
	Java Executable	f:\apps\jdk1.3\bin\java.exe	Path to your JVM executable
	System.out Log File	{jrun.rootdir}/logs/{jrun.server.name}-out.log	Location where System.out messages appear
	System.err Log File	{jrun.rootdir}/logs/{jrun.server.name}-err.log	Location where System.err messages appear
	JRun Control Port	53000	Port used by JRun to send server commands
	Classpath	{jrun.rootdir}/servers/lib	Additional classpath entries
	Java Arguments	-Djava.compiler=none	Additional command-line arguments passed to the Java Executable
	Library Path	{servlet.jnipath};{ejb.jnipath}	Directory of native JNI

2. In the right pane, click Edit. The Java Settings edit window appears.

3. Edit the properties as described in the following table.

JVM Properties

Property	Description
Java Executable	Enter the path to your Java Virtual Machine.If you change your JVM, you may also need to change the command-line arguments in the `Java Arguments` field.
System.out Log File	Enter the full path name where the JVM's `system.out` messages are logged.
System.err Log File	Enter the full path name where the JVM's `system.err` messages are logged.
JRun Control Port	Enter a unique port number. The default value of this port is determined by the JRun installation script. JRun uses this port for status and shutdown information.
Java Classpath (java.exe only)	A classpath is a list of directories that a Java process searches to find classes. You can add your classes to this path or store your classes in the `<JRun_directory>/classes` directory which JRun adds to the classpath by default. Enter additional paths to the Java classpath in the text field. This Java classpath is used by your servlets within the current JRun server. For more information on classpaths, refer to *Developing Applications with JRun*
Classpath	Enter the locations of the classes and `.jar` files that JRun itself requires to run. If you enter a directory, all `.jar` files in that directory will be included in the classpath.
Java Arguments	Enter any command-line arguments passed by JRun to your JVM executable when JRun starts the JVM.
Library Path	Enter the directory containing a Java Native Interface (JNI) if you want to use statements in another programming language (such as C or C++) in your servlets. You can specify multiple directories separated by semicolons (Windows) or colons (UNIX).

4. To apply your changes, click `Update`.
5. Restart the JRun server.

Using Java command-line options

By default, JRun for the Windows and Solaris platforms uses JRE 1.2. Before using the editor to specify another JVM in the Java Executable field, you should identify each of the available command-line options. (You can pass additional command-line options in the `Java Arguments` field.) Following are the command-line options for two popular JVMs.

Sun Microsystems JDK1.1.7b Options

```
Sun Microsystems JDK1.1.7b

usage: java [-options] class

where options include:
    -help             print out this message
    -version          print out the build version
    -v -verbose       turn on verbose mode
    -debug            enable remote JAVA debugging
    -noasyncgc        don’t allow asynchronous garbage collection
    -verbosegc        print a message when garbage collection occurs
    -noclassgc        disable class garbage collection
    -ss<number>       set the maximum native stack size for any thread
    -oss<number>      set the maximum Java stack size for any thread
    -ms<number>       set the initial Java heap size
    -mx<number>       set the maximum Java heap size
    -classpath <directories separated by semicolons>
        list directories in which to look for classes
    -prof[:<file>]    output profiling data to .\java.prof or .\<file>
    -verify           verify all classes when read in
    -verifyremote     verify classes read in over the network [default]
    -noverify         do not verify any class
    -nojit            disable JIT compiler

Sun Microsystems JDK1.2 (Java 2 Platform)

Usage: java [-options] class [args...]
        (to execute a class)
    or  java -jar [-options] jarfile [args...]
        (to execute a jar file)

where options include:
    -cp -classpath <directories and zip/jar files separated by ;>
        set search path for application classes and resources
    -D<name>=<value>
        set a system property
    -verbose[:class|gc|jni]
        enable verbose output
    -version  print product version
    -? -help  print this help message
    -X print help on non-standard options
```

Microsoft Command-Line Loader Options

```
Microsoft (R) Command-line Loader for Java  Version 5.00.3155
Copyright (C) Microsoft Corp 1996-1999. All rights reserved.

Usage: JView [options] <classname> [arguments]

Options:
    /?                  displays usage text
    /cp <classpath>     set class path
    /cp:p <path>        prepend path to class path
    /cp:a <path>        append path to class path
    /n <namespace>      namespace in which to run
    /p                  pauses before terminating if an error occurs
    /v                  verify all classes
    /d:<name>=<value>   define system property
    /a                  execute AppletViewer
    /vst               print verbose stack traces (requires debug classes)

Classname:
    .CLASS file to be executed.

Arguments:
    command-line arguments to be passed on to the class file
```

Configuring JRun server event logs

The JRun logging mechanism allows you to control the content of the log files for each JRun server. This can be helpful for troubleshooting and load-balancing your Web server. This section explains how to configure the event logs for JRun servers using the JMC.

On UNIX systems, you can also redirect the `Java.System.err` and `java.System.out` output to the console using the `-console` option of the `jrun` command. For more information, refer to “Using the JRun command” on page 86.

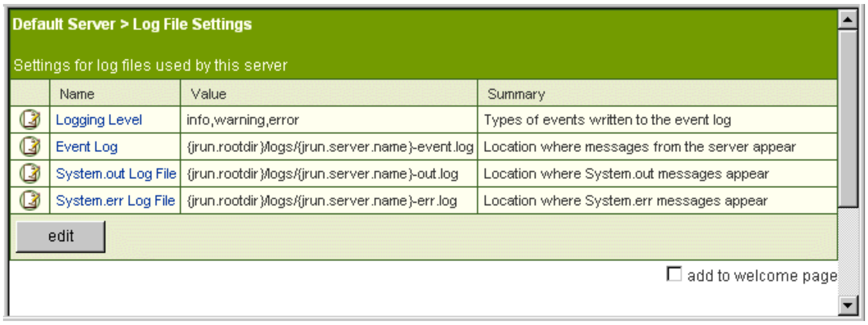
For information on configuring event logs for the JVMs, refer to “Configuring Java Virtual Machines” on page 91. For information on configuring event logs for JRun applications, refer to “Configuring JRun application event logs” on page 121.

For additional information on using the JRun logging mechanism, refer to *Developing Applications with JRun*.

To edit log settings for a JRun server:

1. In the left pane, select *`machine_name`* `>` *`JRun_server_name`* `> Log File Settings`.

The Log File Settings panel appears.



Default Server > Log File Settings

Settings for log files used by this server

	Name	Value	Summary
	Logging Level	info,warning,error	Types of events written to the event log
	Event Log	{jrun.rootdir}/logs/{jrun.server.name}-event.log	Location where messages from the server appear
	System.out Log File	{jrun.rootdir}/logs/{jrun.server.name}-out.log	Location where System.out messages appear
	System.err Log File	{jrun.rootdir}/logs/{jrun.server.name}-err.log	Location where System.err messages appear

2. In the right pane, click Edit. The Log File Settings edit window appears.
3. Enter the properties in the right pane as described in the following table.

JRun Server Event Log Properties

Property	Description
Logging Level	Select each logging level you would like to add to the log files. Info, error, and warning are set by default. The other options are debug and metrics.
Event Log	Set the path and name of the log file. The default is `{jrun.rootdir}/logs/{jrun.server.name}-event.log`.
System.out Log File	Enter the full path name where the JRun server's `system.out` messages are logged.
System.err Log File	Enter the full path name where the JRun server's `system.err` messages are logged.

4. Click `Update` to apply your changes.
5. Restart your JRun server.

Configuring JDBC Data Sources

Using the JMC you can add, remove, and test JDBC-compliant data sources used by JRun. By using this interface to the data source settings, you no longer have to hard-code these settings in your servlets.

Tip Using the JMC, you can change databases, passwords, and other arguments passed to the database without changing every servlet that uses the data source or recompiling those servlets.

The following table contains common JDBC drivers and URLs.

Database	Driver/URL(s)
Sun/Merant JDBC to ODBC Bridge	Driver: sun.jdbc.odbc.JdbcOdbcDriver URL: jdbc:odbc:database_name
Oracle	Driver: oracle.jdbc.driver.OracleDriver URL: jdbc:oracle:thin:@<mc-name>:<port-no>:<sid> URL: jdbc:oracle:oci8:@
MySQL	Driver: org.gjt.mm.mysql.Driver URL: jdbc:mysql://[hostname][:port]/ database_name[?param1=value1][&param2=value2]...
PostgreSql	Driver: postgresql.Driver URL: jdbc:postgresql:database_name URL: jdbc:postgresql://host/database_name URL: jdbc:postgresql://hostport/database_name
Informix	Driver: com.informix.jdbc.IfxDriver URL: jdbc:informix-sqli://{<ip-address>|<host-name>}:<port-number> [/<database_name>]:INFORMIXSERVER=<server-name>; user=<username>;password=<password>
Sybase	jConnect 4.2: Driver: com.sybase.jdbc.SybDriver URL: jdbc:sybase:Tds:<hostname>:<port#>/<database_name> jConnect 5.2: Driver: com.sybase.jdbc2.jdbc.SybDriver URL: jdbc:sybase:Tds:<hostname>:<port#>/<database_name>
DB2	COM.ibm.db2.jdbc.app.DB2Driver jdbc:db2:database_name
Interbase	Driver: interbase.interclient.Driver URL: jdbc:interbase://hostname/database_name
Hypersonic SQL	Driver: org.hsql.jdbcDriver URL: jdbc:HypersonicSQL:database_name

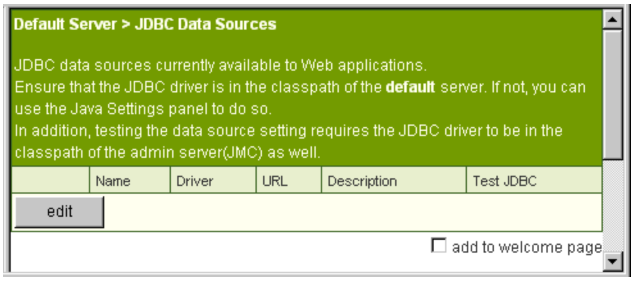
Database	Driver/URL(s)
Domino	Driver: jdbc:domino:<subname> URL: jdbc:domino:/{UNC}
Quadcap Embeddable Database	Driver: com.quadcap.jdbc.JdbcDriver URL: jdbc:qed:database_name
PointBase	Driver: com.pointbase.jdbc.jdbcUniversalDriver URL: jdbc:pointbase:<database_name>

For examples on accessing data sources that are set up in the JMC, refer to *Developing Applications with JRun*.

To configure JDBC data sources:

1. In the left pane, select *`machine_name`* `>` *`JRun_server_name`* `> JDBC Data Sources`.

   The JDBC Data Sources panel appears.



Default Server > JDBC Data Sources

JDBC data sources currently available to Web applications.
Ensure that the JDBC driver is in the classpath of the **default** server. If not, you can use the Java Settings panel to do so.
In addition, testing the data source setting requires the JDBC driver to be in the classpath of the admin server(JMC) as well.

	Name	Driver	URL	Description	Test JDBC

2. In the right pane, click Edit. The JDBC Data Sources edit window appears.
3. Enter the properties in the right pane as described in the following table.

JDBC Data Source Settings

Field	Description
Name	Enter the name of the JDBC data source. This name is used in the servlet's code when establishing a connection to the database. This field is required.
Display Name	Enter the display name for the JDBC data source.
Driver	Enter the class name of the JDBC driver. For example, `sun.jdbc.odbc.JdbcOdbcDriver`. This field is required.

JDBC Data Source Settings

Field	Description
URL	Enter the URL that points to your data source. For example, `jdbc:odbc:`*`fred`*, where *`fred`* is the name of the data source you set up. This field is required.
Description	Enter a description of this JDBC data source. This description is for the JRun JMC only.
Pooling	Select the `Pooling` checkbox to enable connection pooling for your JDBC data source. This is highly recommended to increase performance.
Timeout (min)	Enter a time, in minutes, that JRun allows a data source connection to remain inactive before closing that connection. The default is 30 minutes.
Interval (sec)	Enter a time, in seconds, that JRun waits before closing an expired data source connection. The default is 30 seconds.
Username	Enter a user name if the database requires authentication.
Password	Enter a password corresponding to the `Username` if the database requires authentication.
Vendor Arguments	Enter name/value pairs for vendor-specific arguments. For example, some database vendors allow you to pass arguments that set connection pooling parameters. The format is `name=value`. For more information, refer to your database vendor's documentation.

4. To delete a JDBC data source, select its `Delete?` checkbox.
5. Click `Update` to apply your changes.
6. Test the data source connection by clicking the `Test` button.

Configuring Web Servers

JRun creates an instance of the JRun Web Server (JWS) for each JRun server and links them with a JRun Connection Module (JCM). At installation time, this means that two instances of the JWS are running, one for the `admin` server and one for `default`.

If you add a new JRun server, JRun creates another JWS instance so that you can begin serving requests on this new server immediately. However, most production environments require external Web servers to meet their needs, and JRun provides some control over the connection to these Web servers. This connection is created using the JRun Connector Wizard.

Note Modifying the connection settings for the `admin` JRun server can change the way you access the JRun Management Console. Be sure to make a note of all changes so that they can be undone if necessary.

The following sections explain how to configure the connection between each JRun server and it's JWS or external Web server.

Concurrency overview

Part of configuring the connector between JRun and the JWS or your external Web server is optimizing the concurrency settings. Concurrency defines how HTTP requests are pooled and distributed.

One thing to keep in mind is that "concurrent requests" and "concurrent users" are two distinct concepts. While you might think that you need to support 2000 concurrent *requests*, you might really mean 2000 concurrent *users*, which may actually create only 100 requests at the same time. JRun allocates one thread for each request.

In general, you should not change the default concurrency settings in JRun unless you run a very high volume Internet site. There are four settings in JRun with which you can configure concurrency settings for the connection to your Web server. These settings are:

- Idle Thread Timeout
- Minimum Thread Count
- Maximum Active Requests
- Maximum Concurrent Requests

In an environment where your Web site often experiences spikes in traffic, you should set the Minimum Thread Count higher so that a group of threads do not have to be created when a spike in your Web site traffic occurs. Or you can set the Minimum Thread Count to the expected steady state load of concurrent requests. For example, if you have 200 concurrent requests at all times, then Minimum Thread Count should be set at 200.

If, for example, the average response time of your Web server is slow because of a three-step RMI-CORBA-database transaction, then your site might need to queue up many more requests to maintain throughput without refusing any new requests. In this case, you should increase the Maximum Concurrent Requests to a value greater than the number of expected requests. The Maximum Concurrent Requests setting acts as a safety valve for resources.

Note Before changing any of the default JRun concurrency settings, be sure that the traffic patterns are truly observable. Arbitrarily changing settings can cause resources to be wasted.

You edit the concurrency settings in the Web Server edit windows. For information on editing these settings for the JWS or an external Web server, refer to “Configuring the JWS” on page 101 and “Configuring external Web servers” on page 104 respectively.

Using the JRun Connector Wizard

The JRun Connector Wizard establishes a link between a Web server and a JRun server. This connection is in the form of a JRun Connection Module (JCM). The default installation of JRun includes one JCM that connects the `default` JRun server to a JWS. JRun servers can be connected to any number of Web servers.

Note In most cases, you should select the Default Server to connect an external Web server to. The JRun Admin Server has its own Web server and is used only for administration of your JRun installation. The default server is provided for you to deploy servlets, JSPs, and Web applications. JRun’s demo application runs on the Default Server.

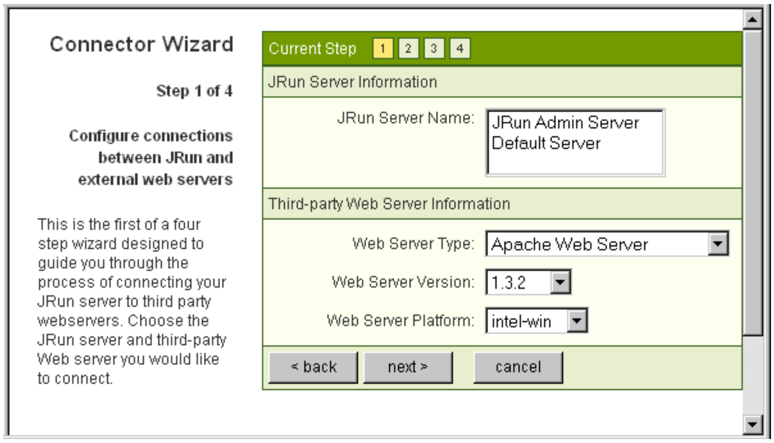
This section provides a general process of how to use the JRun Connector Wizard. For information on how to connect JRun servers to *specific* Web servers, refer to the following sections in Chapter 2:

- “Configuring Apache” on page 38
- “Configuring IIS 3.0/PWS” on page 43
- “Configuring IIS 4.0/5.0” on page 46
- “Configuring Netscape/iPlanet” on page 53
- “Configuring WebSite Pro” on page 61
- “Configuring the Zeus Web Server” on page 70

To use the JRun Connector Wizard:

1. Stop the Web server you want to connect to JRun.
2. Select the `Connector Wizard` link in the access bar.

The JRun Connector Wizard appears in the right pane.



This is the first of a four step wizard designed to guide you through the process of connecting your JRun server to third party webservers. Choose the JRun server and third-party Web server you would like to connect.

JRun Server Information

JRun Server Name: JRun Admin Server, Default Server

Third-party Web Server Information

Web Server Type: Apache Web Server

Web Server Version: 1.3.2

Web Server Platform: intel-win

3. Enter the appropriate information in each step and click Next. If you make a mistake, click Back.

   When you have finished, JRun indicates that you have successfully configured your connection with the Connector Wizard.

4. Start your Web server

5. Restart the JRun server.

6. Test the connection by requesting the demo application on your Web server:

   `http://localhost:80/demo/index.html`

If the demo application runs, you have successfully configured the connection between JRun and your external Web server. If the demo application does not run correctly, refer to "Troubleshooting" on page 73.

Configuring the JWS

JRun sets up two JRun Web Servers (JWS) during the installation process. One instance of the JWS is required, at least initially, because the JMC itself must be accessed using a Web server with a pre-existing connection to the `admin` JRun server. JRun also installs a second instance of the JWS and connects it to the `default` JRun server.

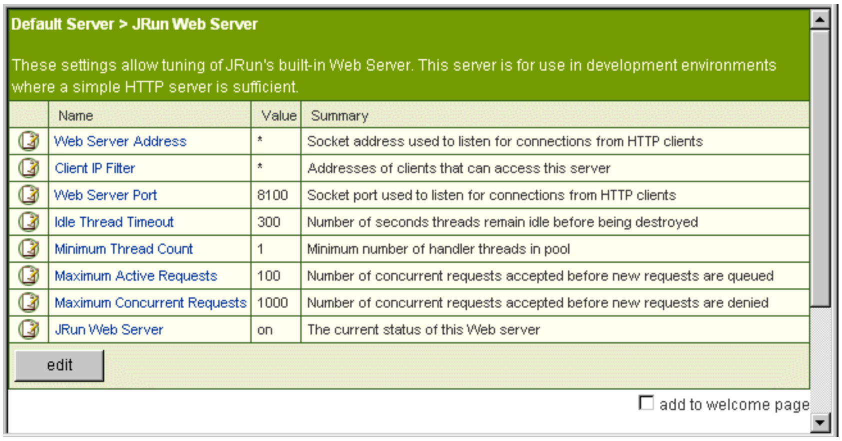
The JWS is a small-footprint, all-Java Web server. Currently, the JWS does not support certain features such as SSL so it may not be adequate for your use in a production environment.

This section describes how to configure the JRun Connection Module for a JWS. If your JRun server is connected to an external Web server, then refer to "Configuring external Web servers" on page 104 to set the endpoint properties for that JCM.

To edit endpoint properties for the JWS:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `JRun Web Server`.

   The JRun Web Server panel appears.



Default Server > JRun Web Server

These settings allow tuning of JRun's built-in Web Server. This server is for use in development environments where a simple HTTP server is sufficient.

Name	Value	Summary
Web Server Address	*	Socket address used to listen for connections from HTTP clients
Client IP Filter	*	Addresses of clients that can access this server
Web Server Port	8100	Socket port used to listen for connections from HTTP clients
Idle Thread Timeout	300	Number of seconds threads remain idle before being destroyed
Minimum Thread Count	1	Minimum number of handler threads in pool
Maximum Active Requests	100	Number of concurrent requests accepted before new requests are queued
Maximum Concurrent Requests	1000	Number of concurrent requests accepted before new requests are denied
JRun Web Server	on	The current status of this Web server

2. In the right pane, click Edit. The JRun Web Server edit window appears.
3. Edit the properties as described in the following table.

JWS Endpoint Properties

Property	Description
Web Server Address	Enter the IP address of the socket listening for connections from HTTP clients on this JWS. If your server has multiple IP addresses (multihoming), enter a list of IP addresses that the JRun server binds itself to on this JWS. The default value is *, which causes this JWS to bind to all server IP addresses.
Client IP Filter	Enter a list of IP addresses to which this JWS will respond. This JWS will ignore requests from all IP addresses *not* specified here. The default value is *, which causes this JWS to respond to requests from all IP addresses.
Web Server Port	Enter a TCP port number. This JWS listens for HTTP requests on this port. The default for the JRun Admin Server's JWS is 8000. The default for the JRun Default Server's JWS is 8100.

JWS Endpoint Properties

Property	Description
Idle Thread Timeout	Enter the number of seconds threads can be idle before JRun destroys them. This parameter determines how quickly the JWS returns to a quiescent state after a busy period. Each time a thread is destroyed, a small amount of system resources is freed up. JRun uses the Java threading mechanism to handle concurrent requests. Instead of creating a new thread for each request, JRun maintains a pool of handler threads that are ready for new requests. This pool of threads grows and shrinks as varying demands are placed on the Web server. Optimally, the parameters for the pool strike a balance between traffic load and the capabilities of the Web server. The default value is 300 seconds.
Minimum Thread Count	Enter the number of threads created in the initial pool (at startup). As the system runs, threads are created and destroyed; however, the pool size will never drop below this minimum number. The default value is 1.
Maximum Active Requests	Enter the upper limit of the number of active concurrent requests that this JWS will handle. Any requests above this limit (up to the Maximum Concurrent Request) will be delayed until a thread is available to service them. This parameter is the JRun's primary mechanism for limiting concurrency on the JWS. The default value is 100.
Maximum Concurrent Requests	Enter the maximum number of concurrent requests that this JWS will either handle or queue for handling. Any requests above this absolute maximum will be dropped. The default value is 1000.
JRun Web Server	Select this checkbox to turn this JWS on. Deselect it to turn this JWS off. Keep in mind that for each JRun server you add, another instance of the JWS is created and connected to that JRun server. While the impact on resources is minimal, you might want to turn off this JWS if you have successfully connected your JRun server to an external Web server using the JRun Connector Wizard. Do not turn off the JWS for the JMC application in the `admin` JRun server.

4. To apply your changes, click `Update`.
5. Restart the JRun server.

Configuring external Web servers

While JRun includes the JWS, you will want to connect JRun servers to external Web servers in a production environment. Connecting an external Web server is described in “Using the JRun Connector Wizard” on page 100, but the end result is that a JRun Connection Module (JCM) manages the connection between the JRun server and an external Web server.

Each JRun server (such as `default`) is connected to a single module, which can be connected to any number of Web servers. Once the connection is established, use the information in this section to fine-tune the JCM.

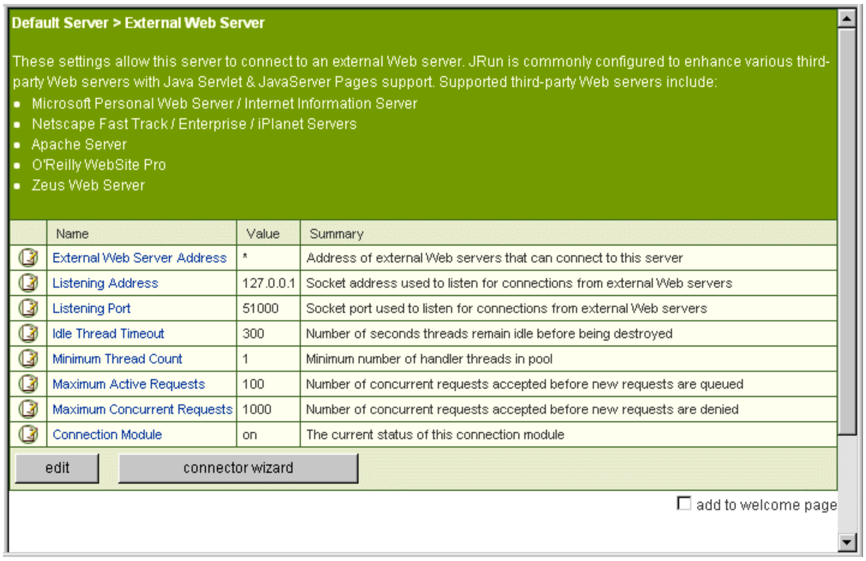
Note From the JMC, you cannot change some of the settings for an external Web server that you can on JRun’s built-in JWS. Consult your Web server’s documentation.

To configure an external Web server’s JCM:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `> External Web Server`.

   Note If you have not yet run the Connector Wizard to connect this JRun server to an external Web server, you will be prompted to do so now.

   The External Web Server panel appears.



Default Server > External Web Server

These settings allow this server to connect to an external Web server. JRun is commonly configured to enhance various third-party Web servers with Java Servlet & JavaServer Pages support. Supported third-party Web servers include:

- Microsoft Personal Web Server / Internet Information Server
- Netscape Fast Track / Enterprise / iPlanet Servers
- Apache Server
- O'Reilly WebSite Pro
- Zeus Web Server

Name	Value	Summary
External Web Server Address	*	Address of external Web servers that can connect to this server
Listening Address	127.0.0.1	Socket address used to listen for connections from external Web servers
Listening Port	51000	Socket port used to listen for connections from external Web servers
Idle Thread Timeout	300	Number of seconds threads remain idle before being destroyed
Minimum Thread Count	1	Minimum number of handler threads in pool
Maximum Active Requests	100	Number of concurrent requests accepted before new requests are queued
Maximum Concurrent Requests	1000	Number of concurrent requests accepted before new requests are denied
Connection Module	on	The current status of this connection module

2. In the right pane, click Edit. The External Web Server edit window appears.
3. Edit the properties as described in the following table.

JRun Connection Module Properties

Property	Description
External Web Server Address	Enter a comma delimited list of IP addresses. This JCM only passes requests from those addresses to the JRun server. The default value is *, which causes this JCM to accept requests from all IP addresses.
Listening Address	Enter the IP address of the socket listening for connections from external Web servers. This feature is useful when your server has multiple IP addresses (multihoming). The default value is *, which causes JRun to bind to all server IP addresses.
Listening Port	Enter a unique port number that this JCM uses to listen for connections from external Web servers. Do not confuse this port with the external Web server's HTTP port.
Idle Thread Timeout	Enter the number of seconds threads can be idle before JRun destroys them. This parameter determines how quickly the Web server returns to a quiescent state after a busy period. Each time a thread is destroyed, a small amount of system resources is freed up. JRun uses the Java threading mechanism to handle concurrent requests. Instead of creating a new thread for each request, JRun maintains a pool of handler threads that are ready for new requests. This pool of threads grows and shrinks as varying demands are placed on the Web server. Optimally, the parameters for the pool strike a balance between traffic load and the capabilities of the Web server. The default value is 300 seconds.
Minimum Thread Count	Enter the number of handler threads created in the initial pool (at startup). As the system runs, threads are created and destroyed; however, the pool size will never drop below this minimum number. The default value is 1.

JRun Connection Module Properties

Property	Description
Maximum Active Requests	Enter the upper limit of the number of concurrent requests that JRun will handle. Any requests above this limit will be delayed until a handler thread is available to service them. This parameter is JRun's primary mechanism for limiting concurrency on the external Web server. The default value is 100.
Maximum Concurrent Requests	Enter the maximum number of requests that the Web server can handle. Any requests above this absolute maximum will be dropped. The default value is 1000.
Connection Module	Select this checkbox to turn this JCM on. Deselect it to turn this JCM off. Turning the Connection Module off severs the connection between JRun and your external Web server. This can result in errors if users try to access JSP pages or servlets.

4. To apply your changes, click `Update`.
5. Restart the JRun server.

Configuring Web Applications

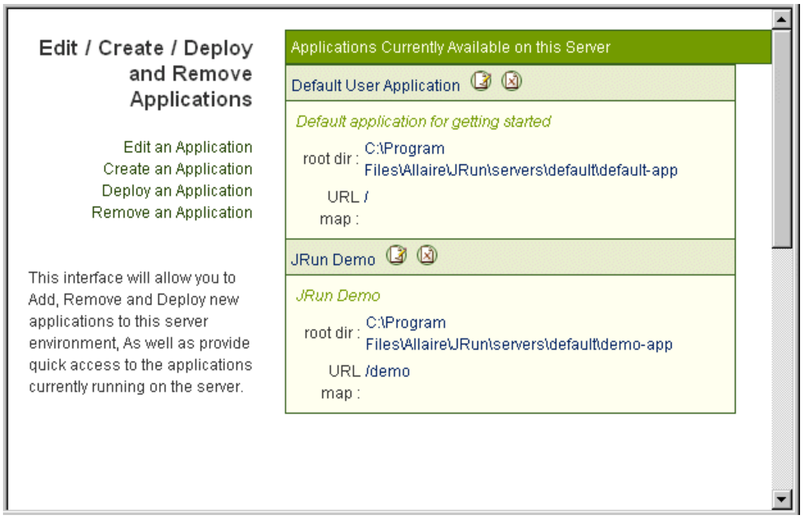
A Web application can consist of servlets, JSPs, static files, and other resources. You place these resources according to a predefined directory structure, such that they can be deployed to any servlet-enabled Web server.

JRun provides two methods for adding applications to a JRun implementation. You can either deploy an existing Web application's Web Application Archive (`.war`) file or create a new application.

You can add any number of Web applications to each JRun server. The default installation includes the following applications:

- `JRun Management Console` application in the `admin` JRun server mapped to `/`.
- `JRun Demo` application in the `default` JRun server mapped to `/demo`.
- `Default User Application` in the `default` JRun server mapped to `/`.

The JMC provides the Application panel that you can access by selecting *`machine_name`* `>` *`JRun_server_name`* `> Web Applications`.


Edit / Create / Deploy and Remove Applications

Edit an Application
Create an Application
Deploy an Application
Remove an Application

This interface will allow you to Add, Remove and Deploy new applications to this server environment, As well as provide quick access to the applications currently running on the server.

Applications Currently Available on this Server

Default User Application

Default application for getting started

root dir : C:\Program Files\Allaire\JRun\servers\default\default-app

URL map : /

JRun Demo

JRun Demo

root dir : C:\Program Files\Allaire\JRun\servers\default\demo-app

URL map : /demo

The JMC provides quick links to the edit and delete functions for each application in the Applications panel.

This section describes the following:

- "Creating applications" on page 108
- "Deploying applications" on page 109
- "Editing applications" on page 111
- "Removing applications" on page 113
- "Mapping application paths" on page 114
- "Creating application hosts" on page 115
- "Adding application parameters" on page 117
- "Changing file settings" on page 118
- "Configuring the JSP compiler" on page 119
- "Configuring JRun application event logs" on page 121
- "Mapping MIME types" on page 122
- "Configuring session tracking" on page 123

For more information on building and deploying applications, refer to *Developing Applications with JRun*.

Creating applications

You can create an empty application and register it with a JRun server using the JMC. This process creates an empty directory structure for the application consisting of the application's root directory, `WEB-INF`, `WEB-INF/classes`, and `WEB-INF/lib` directories.

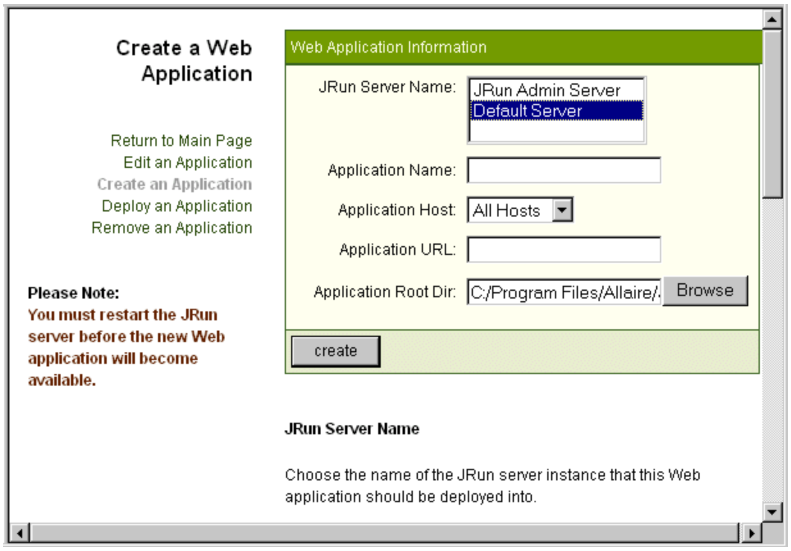
To add an empty application:

1. Select `*machine_name* > *JRun_server_name* > Web Applications`.

   The Application panel appears.

2. Click the Create an Application link.

   The Create a Web Application panel appears.



Create a Web Application

Return to Main Page
Edit an Application
Create an Application
Deploy an Application
Remove an Application

Please Note:
You must restart the JRun server before the new Web application will become available.

Web Application Information

JRun Server Name:	JRun Admin Server / Default Server (selected)
Application Name:	
Application Host:	All Hosts
Application URL:	
Application Root Dir:	C:/Program Files/Allaire/.

JRun Server Name

Choose the name of the JRun server instance that this Web application should be deployed into.

3. Edit the properties as described in the following table.

Creating Web Applications

Field	Description
JRun Server Name	Select a JRun server to deploy this Web application into.
Application Name	Enter a name for the new application. You cannot have more than one application with the same name within the same JRun server. This name appears in the JMC and log files.

Creating Web Applications	
Field	**Description**
Application Host	If you are implementing your application in a multi-homing environment, select a host from the drop-down list. Otherwise, select All Hosts (the default). For more information on application hosts, refer to "Creating application hosts" on page 115.
Application URL	Enter the URL prefix that clients use to access this Web application. Do not use the same Application URL for more than one application.
Application Root Dir	Enter the directory that the Web application is deployed to or click the Browse button to open the JRun Directory Reader. This is the document root for serving application files. If this directory structure does not exist, JRun will create it for you. Do not store more than one application in the same root directory because files of the same name will be overwritten. The default is `<jrun_directory>/servers/` `<`*`server_name`*`>`

4. Click Create.
5. Restart your JRun server.

Deploying applications

Using the JMC, you can deploy an application's Web Application Archive (`.war`) file onto an existing JRun server. The `.war` file consists of JSPs, servlets, images, and other supporting files for a Web application, and is arranged in a structured hierarchy as defined by the Servlet 2.2 specification. This structure also includes the deployment descriptor file, `web.xml`.

You can also use this function to register an application that does not have a `.war` file, as long as the application is compliant with the Servlet 2.2 specification. The application must have the following minimum requirements to be deployed:

- A root application directory (such as `/foo-app`)
- A `/foo-app/WEB-INF` directory
- A `web.xml` deployment descriptor in the root application directory

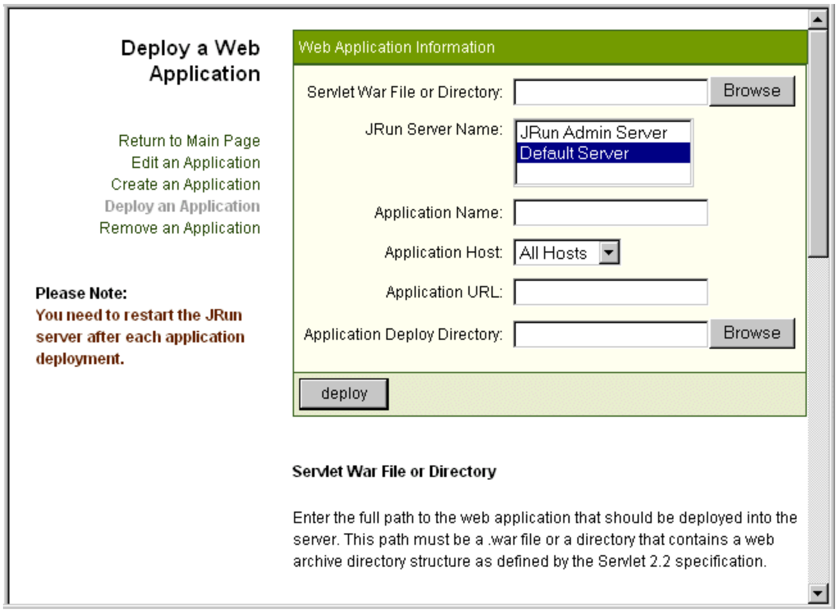
To deploy an application:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Web Applications.`

   The Application panel appears.

2. Click the Deploy an Application link. Alternatively, you can select *`machine_name`* `>` *`JRun_server_name`* and then click the WAR Deployment link at the top of the page.

   The Deploy a Web Application panel appears.



Please Note:
You need to restart the JRun server after each application deployment.

Servlet War File or Directory:
JRun Server Name: JRun Admin Server / Default Server
Application Name:
Application Host: All Hosts
Application URL:
Application Deploy Directory:

Servlet War File or Directory

Enter the full path to the web application that should be deployed into the server. This path must be a .war file or a directory that contains a web archive directory structure as defined by the Servlet 2.2 specification.

3. Enter the properties in the right pane as described in the following table.

New Application Properties

Property	Description
Servlet War File or Directory	Enter the path that the Web application is deployed from or click Browser to use JRun's Directory Browser. For example, `C:\temp\example.war`. You can enter the current location of the application's `.war` file. If you do not have a `.war` file, enter the root of the application's structured directory hierarchy. This hierarchy must be compliant with the Servlet 2.2 specification.
JRun Server Name	Select the JRun server you want to deploy this application into.
Application Name	Enter a name for the new application. You cannot have more than one application with the same name within the same JRun server. This name appears in the JMC and log files.

New Application Properties

Property	Description
Application Host	If you are implementing your application in a multi-homing environment, select a host from the drop-down list. Otherwise, select All Hosts (the default). For more information on application hosts, refer to “Creating application hosts” on page 115.
Application URL	Enter the URL prefix that clients use to access this Web application. Do not use the same Application URL for more than one application.
Application Deploy Directory	Enter the directory that the Web application is deployed to or click the Browse button to open the JRun Directory Reader. This is the document root for serving application files. If this directory structure does not exist, JRun will create it for you. It is not recommended that you store more than one application in the same root directory because files of the same name will be overwritten. The default is `<jrun_directory>/servers/<`*`server_name`*`>`

4. Click Deploy.

   If you are using WebSite Pro as an external Web server connected to JRun, you must set up the mapping for the new application in the WebSite Server Properties application. This mapping is the same as the `/servlet` mapping. For more information, refer to “Mapping a URL prefix to run servlets” on page 61. You do not have to explicitly configure this mapping for other Web servers.

5. Restart your JRun server. The new application appears in the left pane of the JMC under *`JRun_server`* `> Web Applications`.

Editing applications

You can use the JMC to change application settings once they have been deployed. For example, you might want to map an application to multiple hosts, change the application’s root directory, or map a new URL to the application.

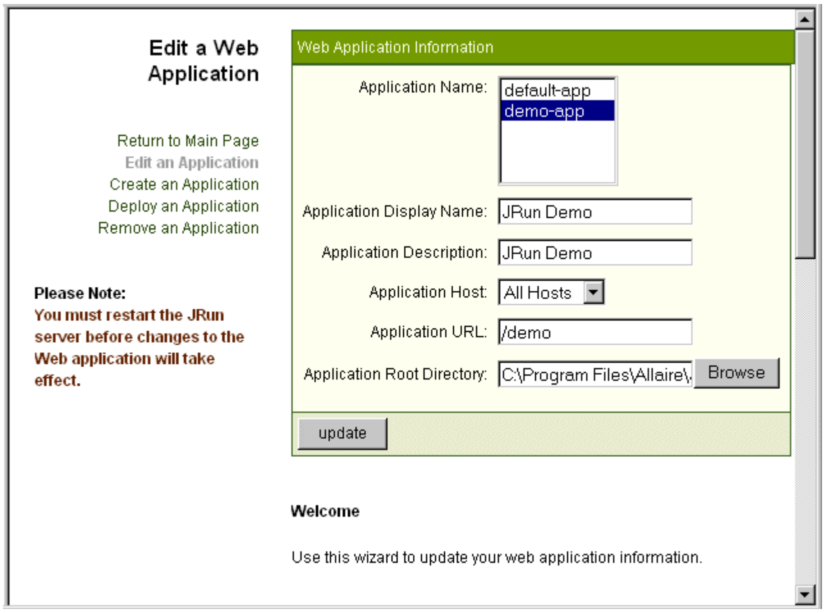
To edit application settings:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Web Applications`.

   The Application panel appears.

2. Click the Edit an Application link.

The Edit a Web Application panel appears.



Edit a Web Application

Please Note:
You must restart the JRun server before changes to the Web application will take effect.

Web Application Information

Application Name: default-app, demo-app

Application Display Name: JRun Demo

Application Description: JRun Demo

Application Host: All Hosts

Application URL: /demo

Application Root Directory: C:\Program Files\Allaire\

Welcome

Use this wizard to update your web application information.

3. Select an application in the `Application Name` field. JRun fills in the remaining fields with that application's current settings.

4. Change the properties in the right pane as described in the following table.

Editing Application Properties

Field	Description
Application Display Name	Change the name for the application. You cannot have more than one application with the same name within the same JRun server. This name appears in the JMC and log files.
Application Description	Enter or change the description for the application that will help you identify this application. This field is optional.
Application Host	If you are implementing your application in a multi-homing environment, select a host from the drop-down list. Otherwise, select All Hosts (the default). For more information on application hosts, refer to "Creating application hosts" on page 115.

Editing Application Properties	
Field	**Description**
Application URL	Change the URL prefix that clients use to access this Web application. Do not use the same Application URL for more than one application within an application host.
Application Root Directory	Change the directory that the Web application is deployed to or click the Browse button to open the JRun Directory Reader. This is the document root for serving application files. It is not recommended that you store more than one application in the same root directory because files of the same name will be overwritten.

5. Click Update to apply your changes.
6. Restart the JRun server.

Removing applications

You can remove any existing application from a JRun server using the JMC.

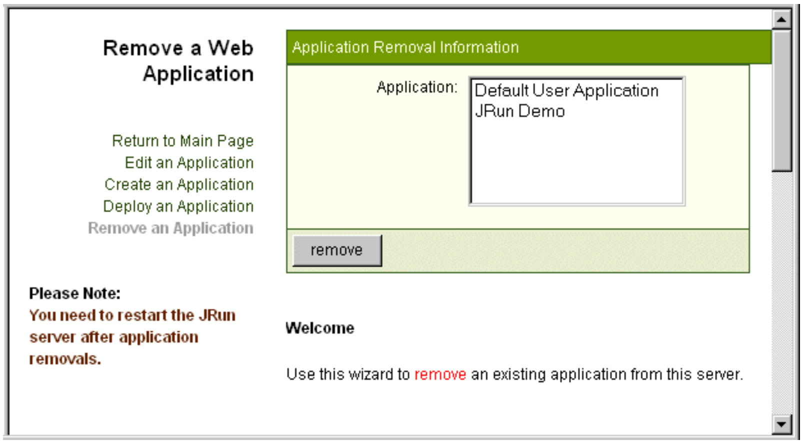
Note Removing an application in the JMC unregisters the application but does not remove any files associated with that application.

To remove an application:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Web Applications`.

   The Application panel appears.
2. Click the Remove an Application link.

The Remove a Web Application panel appears.


Remove a Web Application

Please Note:
You need to restart the JRun server after application removals.

Application Removal Information

Application: Default User Application
JRun Demo

Welcome

Use this wizard to remove an existing application from this server.

3. Select the application you want to remove from the list of available applications in the `Application` listbox.
4. Click Remove.
5. Restart your JRun server.

Mapping application paths

JRun allows you to create mappings that convert part of an application's URL to a real directory. For example, you could create a mapping that converts the virtual path `/foo` to the real path `c:/mydocs/temp` on your hard drive. This mapping can reduce the length of a URL and hide your internal directory structure from a client. In this example, a user requesting a file stored in `c:/mydocs/temp` would enter the URL `http://www.yourdomain.com/foo/<`*`document_name`*`>`.

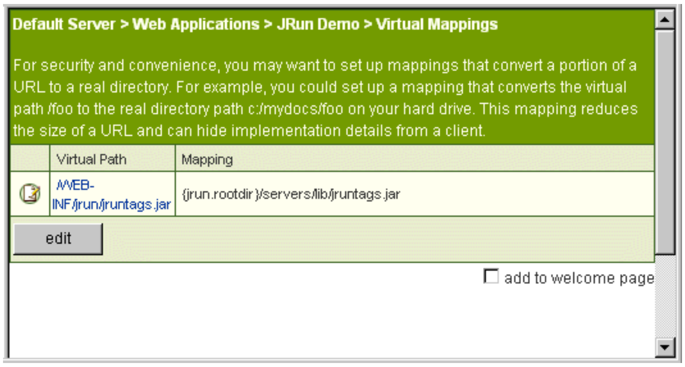
Note There are some pre-existing virtual mappings used internally by JRun to reference external resources. Do not change or remove these mappings.

This section describes how to add your own mappings in the JMC.

To edit application paths:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` *`application_name`* `> Virtual Mappings`.

The Virtual Mappings panel appears.



Default Server > Web Applications > JRun Demo > Virtual Mappings

For security and convenience, you may want to set up mappings that convert a portion of a URL to a real directory. For example, you could set up a mapping that converts the virtual path /foo to the real directory path c:/mydocs/foo on your hard drive. This mapping reduces the size of a URL and can hide implementation details from a client.

	Virtual Path	Mapping
	/WEB-INF/jrun/jruntags.jar	{jrun.rootdir}/servers/lib/jruntags.jar

2. In the right pane, click Edit. The Virtual Mappings edit window appears.
3. In the `Virtual Path` field, enter the portion of a URL mapped to a real path (for example, `/foo`).
4. In the `Real Path` field, enter the path that JRun substitutes for the `Virtual Path` in a URL (for example, `c:/mydocs/temp`). You can use JRun's variables in this field.
5. To delete a path, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart the JRun server for your changes to take effect.

Creating application hosts

When running a Web site, a common technique is to set up multiple Web site host names (like `www1.company.com`, `www2.company.com`) or host multiple domains (`www.yourdomain.com`, `www.mydomain.com`) on a single Web server with a single IP address. This process is called multi-homing or virtual hosting.

Web applications present some problems in a multi-homing environment. According to the servlet specification you can only associate one Web application with one Web site host name. Furthermore, servlets must be able to use their own context information to reference other applications within the same host.

Since you may want to be able to invoke the same application using different DNS host names, JRun introduces the concept of *application hosts*. An application host maps a single application to a set of DNS hosts that can access that application.

Application hosts allow you to map each application to as many Web site host names as you like. You only need to do this in a multi-homing environment, where you have multiple Web site host names mapped to the IP address of a single Web server. If you are not using multi-homing, you do not need to create any application hosts and can use the default for the application host when deploying an application.

Creating and using application hosts for virtual hosts is a multi-step process:

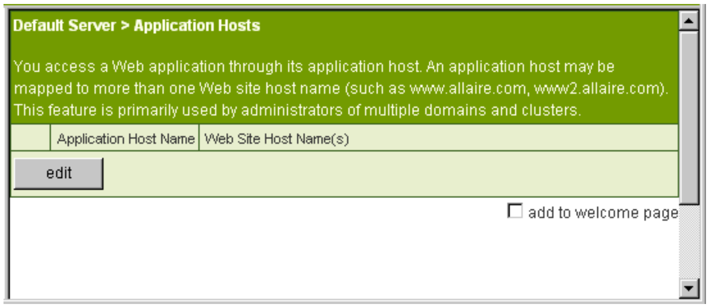
1. Configure your DNS entries and Web servers to support multi-homing. Consult your Web server's documentation for more information.
2. Create an application host and assign any number of Web site host names to it. The procedure for this is below.
3. When creating or deploying an application, choose that application host for an application. This defines the set of DNS names that can be used to access that application. For more information, refer to "Creating Applications" on page 87 or "Deploying Applications" on page 88, respectively.

Using the Application Hosts edit window in the JMC, bind multiple hosts to a single application (through its application host) so that when you access a Web server from one host, you will only see applications that are bound to that host. This section describes how to create a new application host.

To create an application host:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `> Application Hosts`.

   The Application Hosts panel appears.



Default Server > Application Hosts

You access a Web application through its application host. An application host may be mapped to more than one Web site host name (such as www.allaire.com, www2.allaire.com). This feature is primarily used by administrators of multiple domains and clusters.

	Application Host Name	Web Site Host Name(s)

2. Click Edit. The Application Host edit window appears.
3. Enter the name of the application host in the `Application Host` field. This name must be unique and contain no spaces or special characters.
4. Enter a comma-delimited list of hosts in the `Web Site Host Names` field.

   You do not have to assign *fully qualified* host names to an application host. For example, if your application is going to be served up on an internal network, you could map the application host to `fred1` and `fred2`; or if the application is accessible both internally and externally, you can assign `fred1.allaire.com`, `fred2.allaire.com`, `fred1`, and `fred2`.
5. To delete an application host, select its `Delete?` checkbox.
6. Click Update to apply your changes.
7. When creating or deploying an application, select this new host from the `Application Hosts` drop-down list. For more information, refer to "Creating applications" on page 108 or "Deploying applications" on page 109.

8. If you have an existing application, use the Edit an Application edit window to select the new host. For more information, refer to “Editing applications” on page 111.
9. Restart your JRun server.

Adding application parameters

JRun allows you to specify application parameters at runtime by using the JMC. These are accessible by your servlets with the `ServletContext.getInitParameter()` method. These variables are passed to *all* servlets within the Web application and are available in the servlets’ `init()` methods. Initialization parameters cannot be changed until the servlet is reloaded.

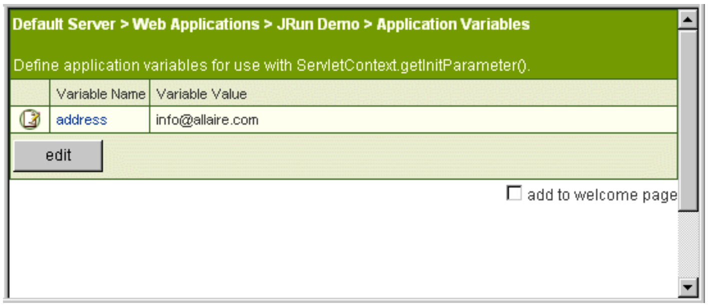
Note If you want to pass initialization parameters to a *single* servlet, use the `Init Arguments` field in the Servlet Definitions edit window. For more information, refer to “Defining servlets” on page 125.

For examples of using initialization parameters, refer to *Developing Applications with JRun*.

To add or modify application variables:

1. Select *`machine_name`* `>` *`JRun_server_name`* `>` *`application_name`* `>` `Web Applications > Application Variables`.

   The Application Variables panel appears.



Default Server > Web Applications > JRun Demo > Application Variables

Define application variables for use with ServletContext.getInitParameter().

	Variable Name	Variable Value
	address	info@allaire.com

2. Click Edit. The Application Variables edit window appears.
3. To add a new application variable, enter a `Variable Name` and it’s associated `Variable Value` in the fields provided. For example, enter `address` in the `Variable Name` field and `info@allaire.com` in the `Variable Value` field.
4. To delete an application variable, select its `Delete?` checkbox.
5. To apply your changes, click `Update`.
6. Restart your JRun server.

Now, calling `ServletContext.getInitParameter("address")` within your servlet code will return the value `info@allaire.com`.

Changing file settings

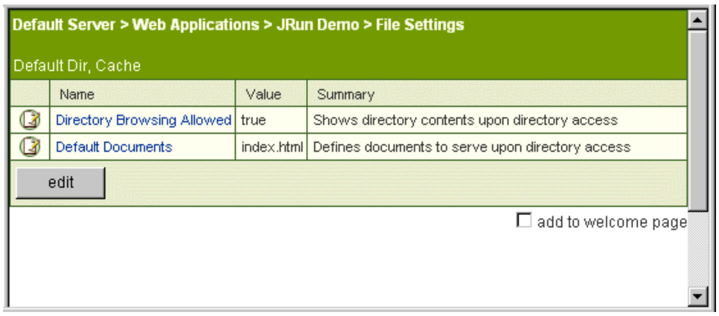
JRun's file settings control the sequence of default document names that a JRun application uses when a document name is not specified in a URL. JRun also enables you to control the browsing of directories.

This section shows you how to change file settings in the JMC.

To edit file settings for a JRun application:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> File Settings`.

   The File Settings panel appears.



Default Server > Web Applications > JRun Demo > File Settings

Default Dir, Cache

	Name	Value	Summary
	Directory Browsing Allowed	true	Shows directory contents upon directory access
	Default Documents	index.html	Defines documents to serve upon directory access

2. In the right pane, click Edit. The File Settings edit window appears.

3. Edit the properties as described in the following table.

File Properties	
Property	**Description**
Directory Browsing Allowed	Select the checkbox (`true`) to display a directory listing when a requested file is not present and no default document is found in a directory. Deselect the checkbox (`false`) to have the server display a "file not found" error in the browser when no default document is found. The default value is `true`.
Default Documents	Enter a comma-delimited list of default pages that the application uses when a page is not specified in a URL. The order of the list matters. For example, enter `index.jsp`, `index.html` to have JRun first try to serve up `index.jsp` and then check for `index.html` when a user makes a request without a page in the URL.

4. To apply your changes, click `Update`.
5. Restart the JRun server.

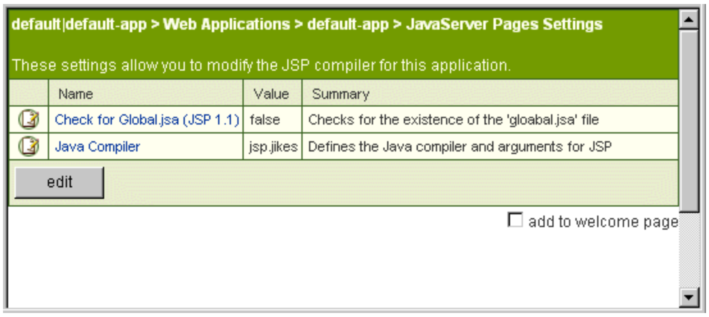
Configuring the JSP compiler

JRun supports JavaServer Pages (JSP), including the industry-standard JSP 1.1 specification. You can modify the JSP settings at the application level in JRun using the information provided in this section. For more information on the JSP compiler, refer to *Developing Applications with JRun*.

To edit JSP properties for an application:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> JavaServer Pages`.

The JavaServer Pages Settings panel appears.



default|default-app > Web Applications > default-app > JavaServer Pages Settings

These settings allow you to modify the JSP compiler for this application.

	Name	Value	Summary
	Check for Global.jsa (JSP 1.1)	false	Checks for the existence of the 'gloabal.jsa' file
	Java Compiler	jsp.jikes	Defines the Java compiler and arguments for JSP

2. In the right pane, click Edit. The JavaServer Pages Settings edit window appears.
3. Edit the properties as described in the following table.

JSP Compilation Properties

Property	Description
Check for Global.jsa (JSP 1.1)	Select the checkbox (`true`) if you want JRun to search for a `global.jsa` file when processing JSPs. If `true`, JRun searches the same directory as the JSP file when the Web server receives the first request for any JSP file in a directory. For more information about the `global.jsa` file, refer to *Developing Applications with JRun.* The default value is `false` (unchecked).

Java Compiler	Enter the path to an external Java compiler to compile your JSPs or leave blank to use JRun's in-process compilation. If you want to use another compiler, be sure to enter an appropriate Java compilation string. For example:
	
	`D:\jdk1.1.7b\bin\javac -nowarn -classpath %c -d %d %f`
	
	or
	
	`jvc /cp:c %c /dest: %d %f`
	
	where
	
	%c = classpath (java classpath to use)
	
	%d = codepath (where to place compiled class files)
	
	`%f = filename`
	
	The default is:
	`{jrun.rootdir}/bin/jikesw -classpath %c -d %d %f`.
	
	For more information, refer to *Developing Applications with JRun*.

4. To apply your changes, click `Update`.
5. Restart your Web server.

Configuring JRun application event logs

The JRun logging mechanism allows you to control the content of the log files. Each application writes out to log files so that you can diagnose errors and maintain the application.

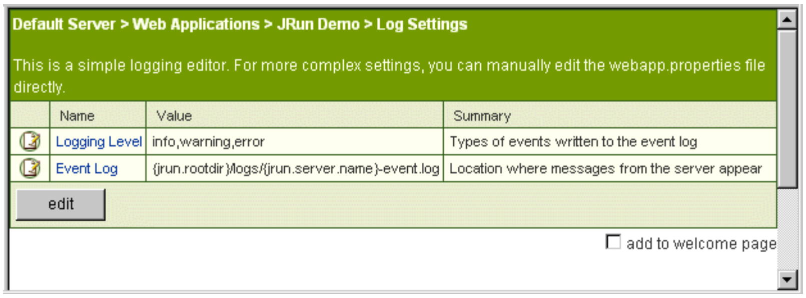
This section explains how to configure the event logs for JRun applications. For information on configuring event logs for JRun servers, refer to “Configuring JRun server event logs” on page 94.

For more information on configuring the JRun logging mechanism, refer to *Developing Applications with JRun*.

To configure the application event logs:

1. In the left pane, select ***machine_name*** `>` ***JRun_server_name*** `>` `Web Applications >` ***application_name*** `> Log File Settings`.

   The application-specific Log File Settings panel appears.



Default Server > Web Applications > JRun Demo > Log Settings

This is a simple logging editor. For more complex settings, you can manually edit the webapp.properties file directly.

	Name	Value	Summary
	Logging Level	info,warning,error	Types of events written to the event log
	Event Log	{jrun.rootdir}/logs/{jrun.server.name}-event.log	Location where messages from the server appear

2. In the right pane, click Edit. The Log Settings Editor appears.
3. Enter the properties in the right pane as described in the following table.

JRun Application Event Log Properties

Property	Description
Logging Level	Select each logging level you would like to add to the log files. Info, error, and warning are set by default. The other options are debug and metrics.
Event Log	Set the path and name of the log file. The default is `{jrun.rootdir}/logs/{jrun.server.name}-event.log`.

4. To apply your changes, click `Update`.
5. Restart the JRun server.

Mapping MIME types

JRun enables you to create associations between specific file extensions and Multipurpose Internet Mail Extension (MIME) types at the Web application level. This means that by using JRun, you can map a request for a certain file extension within a Web application, such as `.html`, to generate a response in a particular MIME type (such as `plain/text`).

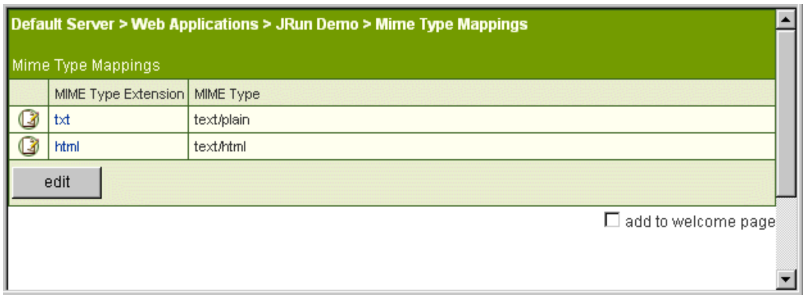
You can also trigger a servlet or chain of servlets based on the MIME type. For more information, refer to “Chaining servlets with MIME type filtering” on page 131.

This section explains how to associate a file extension with a MIME type.

To edit MIME type associations:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> MIME Type Mappings`.

   The MIME Type Mappings panel appears.



Default Server > Web Applications > JRun Demo > Mime Type Mappings

Mime Type Mappings

	MIME Type Extension	MIME Type
	txt	text/plain
	html	text/html

2. In the right pane, click Edit. The MIME Type Mappings edit window appears.
3. In the `MIME Type Extension` field, enter the file extension that you want to associate with a MIME type. For example, `html`.
4. In the `MIME Type` field, enter the MIME type to associate with the extension in the `MIME Type Extension` field.
5. To delete an association, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart your JRun server.

Configuring session tracking

The Servlet 2.2 Specification allows you to use several methods of session tracking with servlets/JSPs or EJBs:

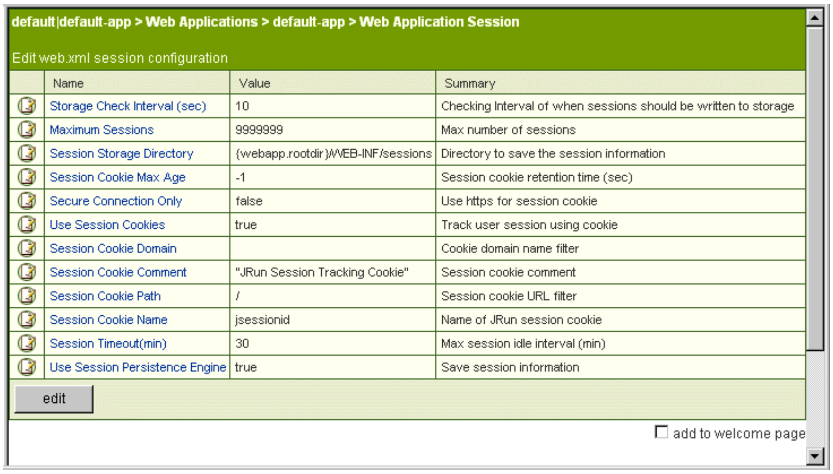
- Cookies
- URL Rewriting
- Hidden form fields

JRun's implementation of the 2.2 Servlet Specification supports all of these methods, but the JMC provides an easy way to configure the most popular of these, cookies. For more information on using the session object in a servlet, refer to *Developing Applications with JRun*.

To edit session tracking properties for an application:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Application >` *`application_name`* `> Session Settings`.

   The Web Application Session panel appears.



default|default-app > Web Applications > default-app > Web Application Session

Edit web.xml session configuration

Name	Value	Summary
Storage Check Interval (sec)	10	Checking Interval of when sessions should be written to storage
Maximum Sessions	9999999	Max number of sessions
Session Storage Directory	{webapp.rootdir}/WEB-INF/sessions	Directory to save the session information
Session Cookie Max Age	-1	Session cookie retention time (sec)
Secure Connection Only	false	Use https for session cookie
Use Session Cookies	true	Track user session using cookie
Session Cookie Domain		Cookie domain name filter
Session Cookie Comment	"JRun Session Tracking Cookie"	Session cookie comment
Session Cookie Path	/	Session cookie URL filter
Session Cookie Name	jsessionid	Name of JRun session cookie
Session Timeout(min)	30	Max session idle interval (min)
Use Session Persistence Engine	true	Save session information

2. In the right pane, click Edit. The Web Application Session edit window appears.

3. Edit the properties as described in the following table.

Session Tracking Properties

Property	Description
Storage Check Interval (sec)	Enter a number, in seconds, that specifies how often sessions should be written to a storage service. The default is 10.
Maximum Sessions	Enter the maximum number of sessions that JRun will track before releasing older sessions. The default is 9999999.
Storage Directory	Enter the full path to the directory where sessions are stored. The default is `{webapp.rootdir}/WEB-INF/sessions`.
Session Cookie Max Age	Enter the number of seconds that a browser retains the JRun cookie used for session tracking. The following numbers have special meanings: `-1` Causes the browser to delete the cookie when the browser exits. `0` Causes the browser to delete the cookie immediately. The default is -1.
Secure Connection Only	Select the checkbox (`true`) to specify that the cookie should only be sent using a secure protocol (https). Use this only when your server supports a secure protocol. The default value is `false`.
Use Session Cookies	Select the checkbox (`true`) to track user sessions using cookies. Deselect the checkbox (`false`) to disable JRun's session tracking. The default value is `true`.
Session Cookie Domain	Enter a single domain name. JRun stores cookies only on hosts that match this domain. For specific implementation details, refer to RFC 2109 *HTTP State Management Mechanism*. By default, this parameter is empty, which indicates that JRun stores cookies on hosts in all domains.
Session Cookie Comment	Enter a comment to appear in the JRun session cookie. Use this comment to specify the cookie's purpose. The default is "`JRun Session Tracking Cookie`".

Session Tracking Properties (Continued)	
Property	**Description**
Session Cookie Path	Enter a limiting URL. JRun only sends session cookies for requests beginning with this URL. A URL referencing the same directory or subdirectory as the one which set the cookie can see the cookie. The default is /.
Session Cookie Name	Enter the name of the JRun session cookie. The default is `jsessionid`.
Session Timeout (min)	Enter the number of minutes that a session is kept alive after its last access (session timeout). The default is 30.
Use Session Persistence Engine	Select the checkbox (`true`) to use Java Serialization to save and restore sessions when JRun shuts down and starts up again. Session data is saved only when JRun is shut down properly. Deselect the checkbox (`false`) to only store session data in the virtual machine. This is a high-availability feature. The default value is `true`.

4. To apply your changes, click `Update`.
5. Restart your Web server.

Configuring Servlets

Web applications can contain any number of servlets that make up the functionality of the application. This section describes how to add servlets to an application and change the settings of those servlets within that application.

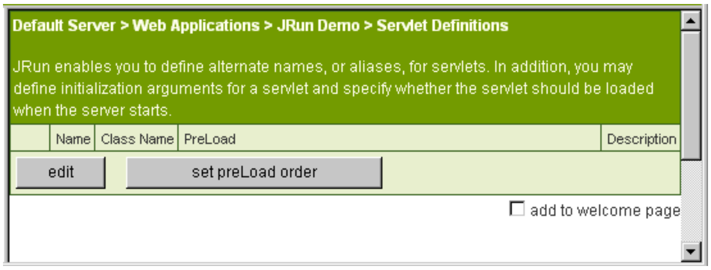
Defining servlets

When adding, or registering, a servlet, you should define it in the JMC and then restart your JRun server so that the servlet is recognized within the context path. The information in this section describes how to use the JMC to add a servlet and modify servlet-specific settings. You can also specify whether the servlet should be loaded when the JRun server starts. By default, servlets are not preloaded.

To define a servlet:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> Servlet Definitions`.

   The Servlet Definitions panel appears.



Default Server > Web Applications > JRun Demo > Servlet Definitions

JRun enables you to define alternate names, or aliases, for servlets. In addition, you may define initialization arguments for a servlet and specify whether the servlet should be loaded when the server starts.

	Name	Class Name	PreLoad	Description

2. In the right pane, click Edit. The Servlet Definitions edit window appears.
3. Enter the properties as described in the following table.

Servlet Configuration Properties

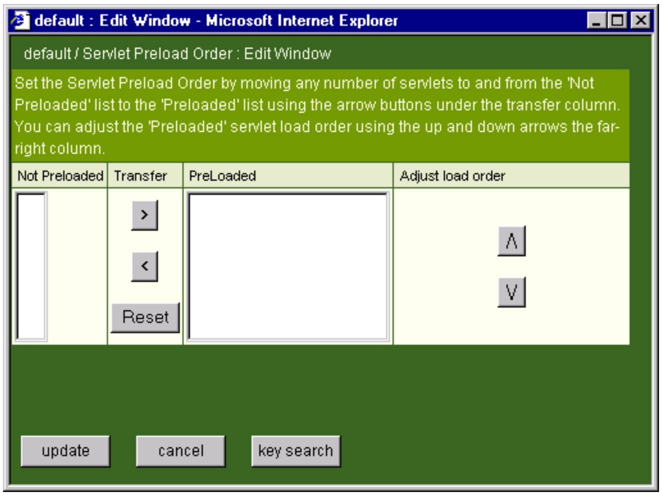
Property	Description
Name	Enter the name of the servlet. This value should not contain spaces or special characters. For example, `DbFuncs`. This field is required.
Class Name	Enter the fully qualified class name of the servlet. For example, if the servlet is named `DbFuncs` and is in a package named `allaire.jrun.rds`, you should enter `allaire.jrun.rds.DbFuncsServlet`. This field is required.
Display Name	Enter the short name for the servlet that appears in the JMC.
Description	Enter a description of the servlet. This field is optional.
Small Icon	Enter the location of a 16x16-pixel icon to represent this servlet. This information is stored in the web.xml file. This is a property that you might use if you have a catalog of servlets and want to represent them each with their own icon. This field is optional.

Servlet Configuration Properties

Property	Description
Large Icon	Enter the location of a 32x32-pixel icon to represent this servlet. This information is stored in the web.xml file. This is a property that you might use if you have a catalog of servlets and want to represent them each with their own icon. This field is optional.
Init Arguments	Enter a list of initialization parameters to pass to this servlet. Note that you can also use the Application Variables Editor to pass the same parameters to *all* servlets within an application. For more information, refer to “Adding application parameters” on page 117. This field is optional.

4. If you do not want to pass any initialization parameters to the servlet, delete the `InitParam=Value` placeholders in the `Init Arguments` field.
5. To delete a servlet definition, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. To indicate whether the servlet should be loaded at server startup, click Set PreLoad Order in the Servlet Definitions panel.

   The Servlet Preload Order dialog box appears.


default : Edit Window - Microsoft Internet Explorer
default / Servlet Preload Order : Edit Window

Set the Servlet Preload Order by moving any number of servlets to and from the 'Not Preloaded' list to the 'Preloaded' list using the arrow buttons under the transfer column. You can adjust the 'Preloaded' servlet load order using the up and down arrows the far-right column.

Not Preloaded
Transfer
PreLoaded
Adjust load order
>
<
Reset
Λ
V
update
cancel
key search

   Use the arrows to move servlets into the Preloaded or Not Preloaded lists and then click the Adjust Load Order arrows to move them up and down within those lists.

8. To apply your changes, click `Update`. JRun assigns each servlet a Preload number, starting with 1. Servlets are preloaded in ascending order.

9. Restart your JRun server.

To change the name of a servlet once it has been modified in the Servlet Definitions edit window, you must delete the entire existing servlet definition and re-add it.

Mapping requests to servlets

Using the JMC, you can map servlets to URL patterns.

When JRun receives an HTTP request, the servlet engine compares the URL pattern against it's defined context paths, then compares the next part of the URL pattern against the registered servlets within that application. Finally, it passes the remaining path information (if any) to the servlet for processing. If the pattern does not match any context paths, then JRun passes the request to the default application.

You can also map file extensions to servlets, and configure a chain of servlets to run one after the other. The following sections describe how to accomplish these tasks using the JMC.

For more information about how JRun serves up files and parses request URLs, refer to *Developing Applications with JRun*.

Mapping servlet URLs

JRun allows you to map servlets to URL prefixes to give you control over how HTTP requests access your servlets. Using the JMC, you can:

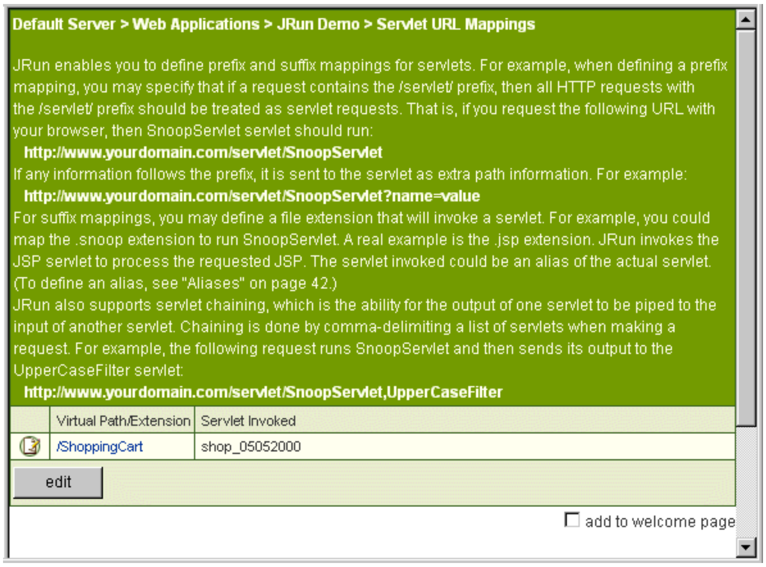
- Map URL patterns to individual servlets or servlet aliases. For example, map requests to `http://www.yourdomain.com/demo` to launch `JRunDemoServlet`.
- Map URL patterns to the JRun `invoker` servlet. For example, if the user requests `http://www.yourdomain.com/ShoppingCart/<`*`anyservlet`*`>` and `/ShoppingCart` is mapped to `invoker`, the value of `<`*`anyservlet`*`>` is passed to the `invoker` servlet to run as a servlet. This is not a recommended approach because it was designed before Web applications became the focus of the servlet specification. Using this type of mapping can cause resource reference problems.

By default, HTTP requests containing the `/servlet` prefix are mapped to the `invoker`, as in `http://www.yourdomain.com/demo/servlet/SimpleServlet`. You can map any number of URLs to a servlet.

To map servlets to URL prefixes:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> Servlet URL Mappings`.

The Servlet URL Mappings panel appears.



Default Server > Web Applications > JRun Demo > Servlet URL Mappings

JRun enables you to define prefix and suffix mappings for servlets. For example, when defining a prefix mapping, you may specify that if a request contains the /servlet/ prefix, then all HTTP requests with the /servlet/ prefix should be treated as servlet requests. That is, if you request the following URL with your browser, then SnoopServlet servlet should run:

http://www.yourdomain.com/servlet/SnoopServlet

If any information follows the prefix, it is sent to the servlet as extra path information. For example:

http://www.yourdomain.com/servlet/SnoopServlet?name=value

For suffix mappings, you may define a file extension that will invoke a servlet. For example, you could map the .snoop extension to run SnoopServlet. A real example is the .jsp extension. JRun invokes the JSP servlet to process the requested JSP. The servlet invoked could be an alias of the actual servlet. (To define an alias, see "Aliases" on page 42.)

JRun also supports servlet chaining, which is the ability for the output of one servlet to be piped to the input of another servlet. Chaining is done by comma-delimiting a list of servlets when making a request. For example, the following request runs SnoopServlet and then sends its output to the UpperCaseFilter servlet:

http://www.yourdomain.com/servlet/SnoopServlet,UpperCaseFilter

	Virtual Path/Extension	Servlet Invoked
	/ShoppingCart	shop_05052000

2. In the right pane, click Edit. The Servlet URL Mappings edit window appears.
3. In the `Virtual Path/Extension` field, enter the URL prefix that invokes a servlet. For example, `/demo` or `/ShoppingCart`. Enter a `'/'` to make the servlet specified in the next field the default servlet for this Web application.
4. In the `Servlet Invoked` field, enter the servlet invoked by the URL prefix. The servlet invoked could be an alias of the actual servlet. To define an alias, see "Aliasing servlets" on page 130.
5. To delete a servlet mapping, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart your JRun server.

Mapping file extensions with suffixes

JRun allows you to map any file extension to any servlet. By default, the `*.jsp` extension is implicitly mapped to the `invoker` servlet, but you can override this with an explicit mapping.

To map file extensions for servlets:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Web Applications >` *`application_name`* `> Servlet URL Mappings`.

   The Servlet URL Mappings panel appears.
2. Click Edit. The Servlet URL Mappings edit window appears.

3. In the `Virtual Path/Extension` field, enter the extension to map to a servlet. Include a wildcard character (*) in the mapping to map all files requested with that extension to the servlet. For example, enter `*.cfm`.
4. In the `Servlet Invoked` field, enter the servlet invoked by the file extension. The servlet can be an alias of the actual servlet. To define an alias, see "Aliasing servlets" on page 130.
5. To apply your changes, click `Update`.
6. Restart your JRun server.

Aliasing servlets

Using the Servlet URL Mappings edit window, you can have an alias reference a servlet. This lets you hide implementation details (such as the real name of a servlet) from users. For example, you might have an alias called `ShoppingCart` reference a servlet whose actual name is `shop_05022000`. When you create a new version of the servlet with a new name, you only have to change the name in one place.

You can use the servlet aliasing technique to reference a chain of servlets as well. For more information, refer to "Chaining servlets with aliases" on page 131.

This section describes how to assign an alias to a single servlet.

To set an alias for a servlet:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `> Web Applications >` *`application_name`* `> Servlet URL Mappings`.

   The Servlet URL Mappings panel appears.
2. In the right pane, click Edit. The Servlet URL Mappings edit window appears.
3. In the `Virtual Path/Extension` field, enter the alias that points to the servlet. For example, enter `ShoppingCart`.
4. In the `Servlet Invoked` field, enter the name of the servlet the alias references. For example, enter `shop_05022000`.
5. To delete a servlet alias, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart your JRun server.

Chaining servlets

JRun supports *servlet chaining*, which is the ability for the output of one servlet to be used as the input of another servlet. The most basic form of servlet chaining is done by adding a comma-delimited list of servlets to the path information when making a request.

For example, the following request runs `Servlet1` and then sends its output to the `Servlet2` servlet:

```
http://www.yourdomain.com/servlet/Servlet1,Servlet2
```

This method of simple chaining may not be adequate for your needs. There are two additional ways to chain servlets in JRun:

- Servlet aliasing
- MIME type filtering

The following sections describe the two methods of servlet chaining in JRun.

Chaining servlets with aliases

In JRun, you can have one name, or alias, reference a list of servlets that form the chain. This is called aliasing.

To chain servlets with aliasing:

1. In the left pane of the JMC, select `*machine_name*` `>` `*JRun_server_name*` `> Web Applications >` `*application_name*` `> Servlet URL Mappings`.

   The Servlet URL Mappings panel appears.
2. In the right pane, click Edit. The Servlet URL Mappings edit window appears.
3. In the `Virtual Path/Extension` field, enter the alias that should invoke the chain of servlets. For example, enter `/Samples`.
4. In the `Servlet Invoked` field, enter a comma-delimited list of servlets in the order in which they should be executed.

   For example, to create a chain that sends the output of `SnoopServlet` to `UpperCaseFilter`, enter `SnoopServlet, UpperCaseFilter`.
5. To delete a servlet chain, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart your JRun server.

Chaining servlets with MIME type filtering

In addition to setting response types, using MIME type mapping also lets you execute servlet chains. Instead of explicitly specifying which servlets to chain based on the request as with aliasing, however, you specify the outgoing MIME type that should trigger a servlet or chain of servlets to run.

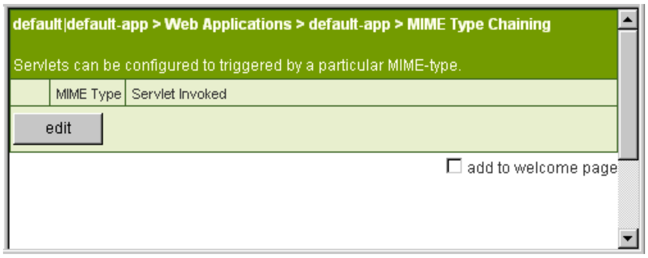
For example, if you map the `text/plain` MIME type to the `UpperCaseFilter` servlet, any servlet that responds with a content type of `text/plain` within that application will be chained to `UpperCaseFilter`. Note that any other type of request responding with content type `text/plain` will *also* trigger the `UpperCaseFilter` servlet.

This process is called *filtering*, since the JRun server filters the output of one response into another servlet or chain of servlets.

To chain servlets with MIME type filtering:

1. In the left pane of the JMC, select `machine_name > JRun_server_name > Web Applications > application_name > MIME Type Chaining`.

   The MIME Type Chaining panel appears.

   default|default-app > Web Applications > default-app > MIME Type Chaining

   Servlets can be configured to triggered by a particular MIME-type.

	MIME Type	Servlet Invoked

2. In the right pane, click Edit.

   The MIME Type Chaining edit window appears.

3. In the `MIME Type` field, enter the MIME type in the form of `xxx/yyy`. For example:

   ```
   text/vnd.wap.wml
   text/plain
   text/html
   image/gif
   image/jpg
   ```

4. In the `Servlet Invoked` field, enter the name of the servlet (or an alias) to be triggered by that MIME type. You can also enter a chain of servlets in this field, separated by commas (for example, `UpperCaseFilter, SpellCheckFilter`).
5. To delete a MIME filter, select its `Delete?` checkbox.
6. To apply your changes, click `Update`.
7. Restart your JRun server.

Using SSI taglets

JRun provides the option of using Server-Side Include (SSI) taglets to embed Java servlets in your HTML files. Taglets provide flexibility to define and implement unique tags within SHTML files.

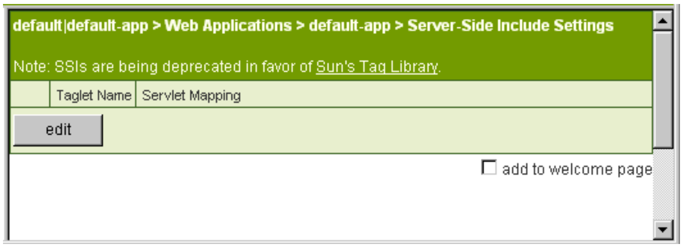
While SSIs were once a common method of creating dynamic content, this functionality in JRun is included primarily to support older implementations. Java Server Pages (JSP) and Java servlet technology have replaced and greatly expanded upon the capabilities of SSI with taglets.

This section explains how to configure SSI taglets for use with JRun. For more information on using SSI taglets, refer to *Developing Applications with JRun*.

To configure Server-Side Includes:

1. In the left pane of the JMC, select *`machine_name`* `>` *`JRun_server_name`* `>` `Web Applications >` *`application_name`* `> Server-Side Includes`.

   The Server Side Includes Settings panel appears.



default|default-app > Web Applications > default-app > Server-Side Include Settings

Note: SSIs are being deprecated in favor of Sun's Tag Library.

	Taglet Name	Servlet Mapping

edit

☐ add to welcome page

2. Click Edit. The Server Side Include Settings edit window appears.
3. Enter the properties as described in the following table.

SSI Properties

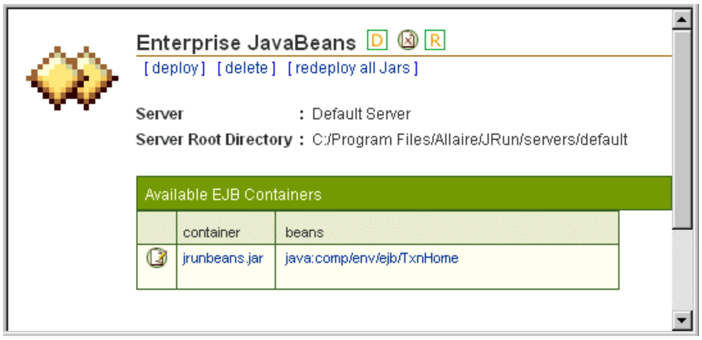
Property	Description
Taglet Name	Enter the taglet name. For example, enter `foo`. When you want to invoke the servlet, use the following code in your Web page: `<foo></foo>`
Servlet Mapping	Enter the servlet referenced by the taglet. For example, enter `SnoopServlet`. Now, when you use the `foo` taglet in your Web page, the `SnoopServlet` servlet will be invoked.

4. To delete an SSI taglet mapping, select its `Delete?` checkbox.
5. To apply your changes, click `Update`.
6. Restart your JRun server.

Configuring Enterprise Applications

JRun 3.0 now supports Enterprise JavaBeans (EJBs) and the deployment of `.ear` files. Once you have developed your EJBs and defined the home and remote interfaces, you are ready for deployment. Note, however, that the term deployment has a slightly different meaning when applied to EJBs in JRun. With servlets, you compile, test, and finally deploy for distribution. With EJBs, you compile, deploy for testing, test, and finally deploy for distribution.

The JMC provides the Enterprise JavaBeans panel that you can access by selecting *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans`.



Server : Default Server
Server Root Directory : C:/Program Files/Allaire/JRun/servers/default

Available EJB Containers

	container	beans
	jrunbeans.jar	java:comp/env/ejb/TxnHome

This section describes the following:

- “Deploying EJBs” on page 134
- “Redeploying EJBs” on page 136
- “Removing EJBs” on page 136
- “Configuring EJBs” on page 137
- “Deploying EAR files” on page 138

For more information on using Enterprise JavaBeans with JRun, refer to *Developing Applications with JRun*.

Deploying EJBs

Use the JMC to prepare beans for deployment with JRun. Deploying in the JMC performs the following actions:

- Generates the home and object implementations for the EJBs listed in the provided `.jar` files.
- Creates stub classes for the generated objects.
- Creates skeletons required for use with JDK 1.1-based clients (only if the `deploy.properties` file specifies `ejipt.isCompatible=true`).
- Prepares the `runtime.properties` file using the properties from `deploy.properties` and the current environment. The `runtime.properties` file is used by JRun to establish the runtime environment.

The JMC deploy tool operates only in the `/deploy` directory. All input (including the `.jar` files) must be available in the `/deploy` directory and all generated output is placed in the `/deploy` directory.

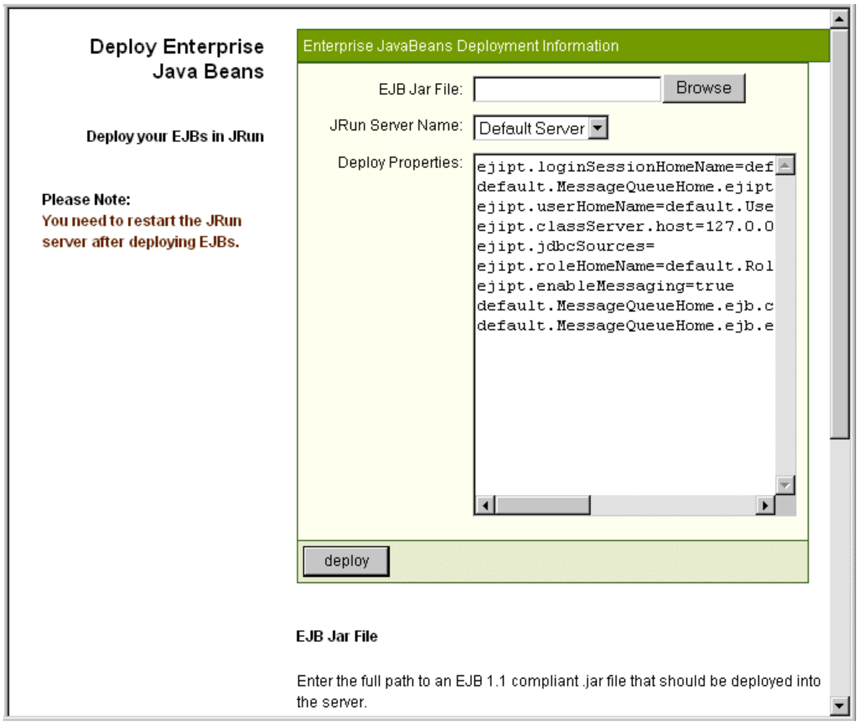
To deploy EJBs:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans`.

The Enterprise JavaBeans panel appears.

2. Click the Deploy link at the top of the page. Alternatively, you can select *`machine_name > JRun_server_name`* and click the EJB Deployment link at the top of the page.

   The Deploy Enterprise JavaBeans panel appears.



Deploy Enterprise Java Beans

Deploy your EJBs in JRun

Please Note:
You need to restart the JRun server after deploying EJBs.

Enterprise JavaBeans Deployment Information

EJB Jar File:
JRun Server Name: Default Server
Deploy Properties:

```
ejipt.loginSessionHomeName=def
default.MessageQueueHome.ejipt
ejipt.userHomeName=default.Use
ejipt.classServer.host=127.0.0
ejipt.jdbcSources=
ejipt.roleHomeName=default.Rol
ejipt.enableMessaging=true
default.MessageQueueHome.ejb.c
default.MessageQueueHome.ejb.e
```

EJB Jar File

Enter the full path to an EJB 1.1 compliant .jar file that should be deployed into the server.

3. Enter the properties in the right pane as described in the following table.

Deploying EJBs	
Field	**Description**
EJB Jar File	Enter the path to the EJB's `.jar` file or click Browse to use JRun's Directory Reader.
JRun Server Name	Select the server into which you want to deploy the EJB.
Deploy Properties	Edit the EJB's server-level deployment properties that are stored in the `deploy.properties` file. You can change, add and delete the `name=value` pairs. When you deploy the EJB, the JMC overwrites the `deploy.properties` file with your changes.

4. Click Deploy.
5. Restart your JRun server.

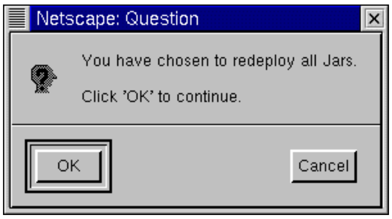
Redeploying EJBs

Any time you change the bean properties either using the JMC's Bean Properties edit window or another tool which modifies the *<bean_name>*`.properties` file, you must redeploy the `.jar` file that contains that EJB. This section describes how to redeploy all `.jars` on a JRun server at once.

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans`.

   The Enterprise JavaBeans panel appears.
2. Click the Redeploy All Jars link at the top of the page.

   JRun prompts you to click OK or Cancel.


Netscape: Question

You have chosen to redeploy all Jars.

Click 'OK' to continue.

Note Redeploying all `.jar` files onto a server may take some time depending on the size and quantity of files.

3. Click OK.

   JRun redeploys all `.jar` files that were previously deployed on this JRun server.

Removing EJBs

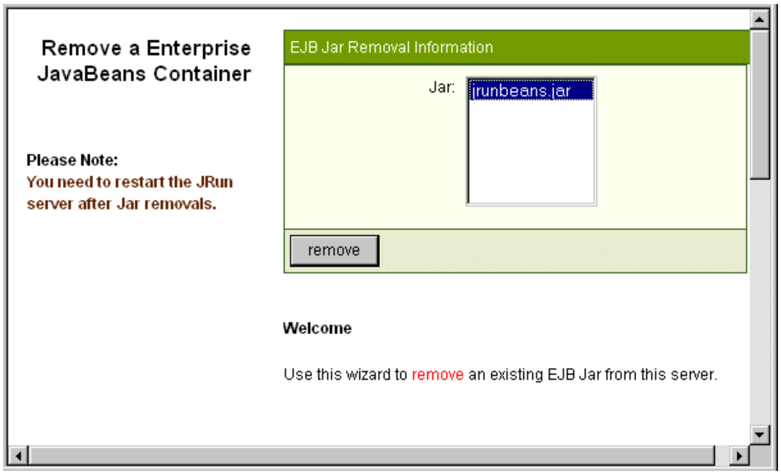
Use this procedure to remove the EJBs in a specified `.jar` file. This does not actually delete the file from the file system. It only unregisters the EJB from the JRun server.

To remove EJBs:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans`.

   The Enterprise JavaBeans panel appears.
2. Click the Delete link at the top of the page.

The Remove an EJB Container panel appears.


Remove a Enterprise JavaBeans Container
EJB Jar Removal Information
Jar:
jrunbeans.jar
remove

Please Note:
You need to restart the JRun server after Jar removals.

Welcome

Use this wizard to remove an existing EJB Jar from this server.

3. Select the `.jar` file you want to remove in the `Application` listbox.
4. Click Remove.
5. Restart your JRun server.

Configuring EJBs

Using the JMC, you can configure the number of available bean contexts that can be managed. The bean context is used to retrieve information about the state of a deployed bean's instance. A context is created at the time a bean instance is created, remains with a bean for the life of the bean instance, and cannot be used by any other bean instance. The context holds information about the bean instance, such as if the instance's state has changed.

The number of available contexts can be managed by setting the `ejipt.maxContexts`, `ejipt.maxFreeContexts` and `ejipt.minFreeContexts` properties in the JMC. The JMC writes this information to the bean's `default.properties` file. The JMC also exposes other bean properties in the Bean Properties edit window. This section describes how to edit those bean properties.

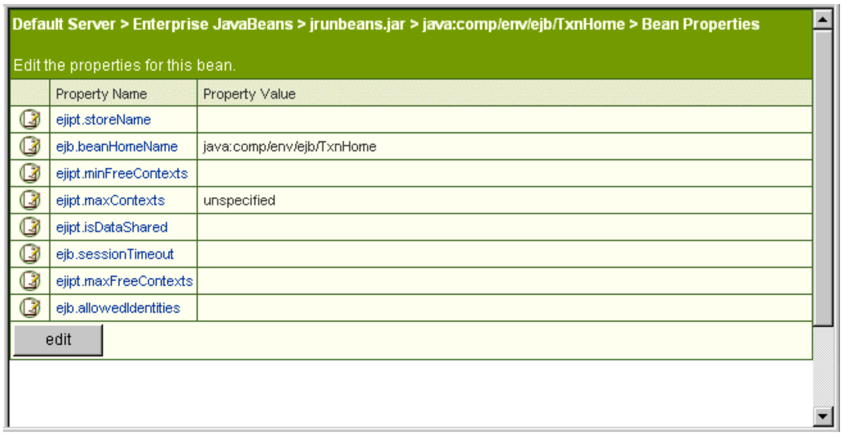
Note If you change an existing bean's properties, you must redeploy the bean's `.jar` file.

For more information on using the bean context properties, refer to the Ejipt Properties API documentation provided with the JRun JavaDocs files, or to *Developing Applications with JRun*.

To configure EJB settings:

1. Select *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans >` *`jar_file`* `>` *`bean`*.

   The Bean Properties panel appears in the right pane.



Default Server > Enterprise JavaBeans > jrunbeans.jar > java:comp/env/ejb/TxnHome > Bean Properties

Edit the properties for this bean.

Property Name	Property Value
ejipt.storeName	
ejb.beanHomeName	java:comp/env/ejb/TxnHome
ejipt.minFreeContexts	
ejipt.maxContexts	unspecified
ejipt.isDataShared	
ejb.sessionTimeout	
ejipt.maxFreeContexts	
ejb.allowedIdentities	

2. Click Edit. The Bean Properties edit window appears.
3. Edit the properties as described in the Ejipt Properties API documentation, provided with the JRun JavaDocs files.
4. Click Update to apply your changes.
5. Select *`machine_name`* `>` *`JRun_server_name`* `> Enterprise JavaBeans`.

   The Enterprise JavaBeans panel appears.
6. Click the Redeploy All Jars link at the top of the page. JRun prompts you to click OK or Cancel.

   Note Redeploying all `.jar` files onto a server may take some time depending on the size and quantity of files.
7. Click OK.

   JRun redeploys all `.jar` files that were previously deployed on this JRun server.
8. Restart your JRun server.

Deploying EAR files

The `.ear` file (Enterprise Application archive) contains the complete directory structure and all files that define the enterprise application. You create an `.ear` file using the same tools that you use to create a `.jar` file. During J2EE application deployment, JRun explodes the `.war` files contained in the `.ear` file, defining new applications in the specified JRun server. JRun also deploys all EJB `.jar` files contained in the `.ear` file.

Use the JMC to install a J2EE application into a specific environment. While installing the `.ear` file, the JMC configures a set of server-specific parameters, populates the directory structure, and updates JRun property files.

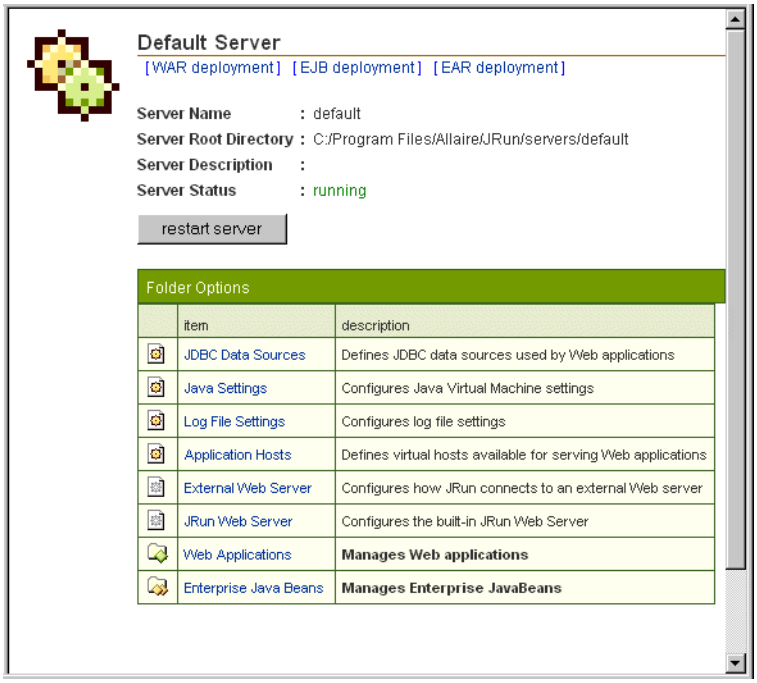
The `.ear` file should contain a `META-INF/application.xml` deployment descriptor, which provides information to the JRun application deployment utilities.

For more information about `.ear` files and J2EE applications, refer to *Developing Applications with JRun*.

To deploy EAR files:

1. Select *`machine_name > JRun_server_name`*.

   The JRun Server panel appears.



Default Server

[WAR deployment] [EJB deployment] [EAR deployment]

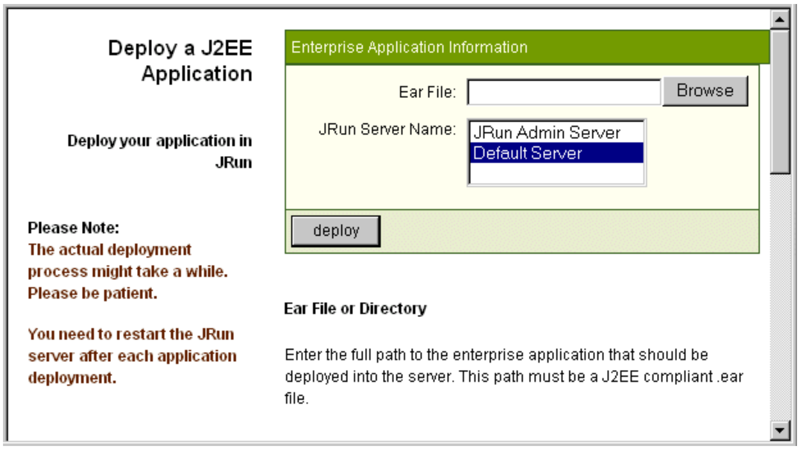
Server Name	: default
Server Root Directory	: C:/Program Files/Allaire/JRun/servers/default
Server Description	:
Server Status	: running

Folder Options

	item	description
	JDBC Data Sources	Defines JDBC data sources used by Web applications
	Java Settings	Configures Java Virtual Machine settings
	Log File Settings	Configures log file settings
	Application Hosts	Defines virtual hosts available for serving Web applications
	External Web Server	Configures how JRun connects to an external Web server
	JRun Web Server	Configures the built-in JRun Web Server
	Web Applications	**Manages Web applications**
	Enterprise Java Beans	**Manages Enterprise JavaBeans**

2. Click the EAR Deployment link.

The Deploy a J2EE Application panel appears.



Deploy a J2EE Application

Deploy your application in JRun

Please Note:
The actual deployment process might take a while. Please be patient.

You need to restart the JRun server after each application deployment.

Enterprise Application Information

Ear File: [ ] Browse

JRun Server Name: JRun Admin Server / Default Server

deploy

Ear File or Directory

Enter the full path to the enterprise application that should be deployed into the server. This path must be a J2EE compliant .ear file.

3. Enter the properties in the right pane as described in the following table.

Deploying .ear Files

Field	Description
EAR File	Enter the path to the `.ear` file or click Browse to use JRun's Directory Reader.
JRun Server Name	Select the JRun server that you want to deploy the `.ear` file into.

4. Click Deploy.
5. Restart your JRun server.

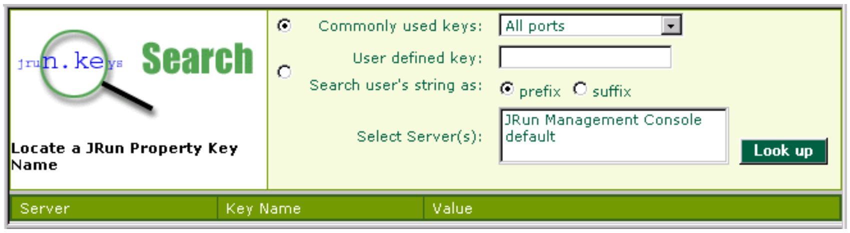
Searching JMC Keys

The Key Search feature allows you to search for keys (property names) by prefix or suffix within one or more JRun servers.

To perform a key search:

1. Click `Key Search` in the access bar. You can also click the `Key Search` button in many JMC panels.

The Key Search window appears in the right pane.



2. To search for a commonly used key, select `Commonly used keys` and select a key from the drop-down listbox.
3. To search for a key you added, select `User defined key` and enter the key in the field provided; then indicate whether the key is a prefix or suffix by selecting the appropriate button.
4. Select the JRun servers you want to search in the `Select Servers(s)` listbox. To select multiple JRun servers, click on the first one and hold the `Ctrl` key down while selecting additional JRun servers.
5. Click Look Up. The lower pane of the window displays the results of the search.

Logging Out

You may need to log out of the JMC for some changes to take effect.

To log out of the JMC:

1. Click `Logout` in the access bar.

   JRun logs you out. The login screen appears.

CHAPTER 4

Property Files

JRun uses property files for initialization and configuration. These files store most of the configurable settings for JRun.

While most common configuration tasks can be accomplished using the JRun Management Console (which writes to the property files), editing the property files can give you a greater understanding of the internal variables available to JRun as well as greater control over some of JRun's settings.

This chapter explains the hierarchical nature of the property files and how you can leverage their accessibility to configure JRun.

Contents

Introduction to Property Files

The property files contain most of JRun's configuration information. JRun reads the property files on startup and stores their values in memory until JRun is restarted. JRun reads the `local.properties` file first, then the rest of the properties files.

The following table describes the property files and their locations in the JRun directory tree.

Property Files

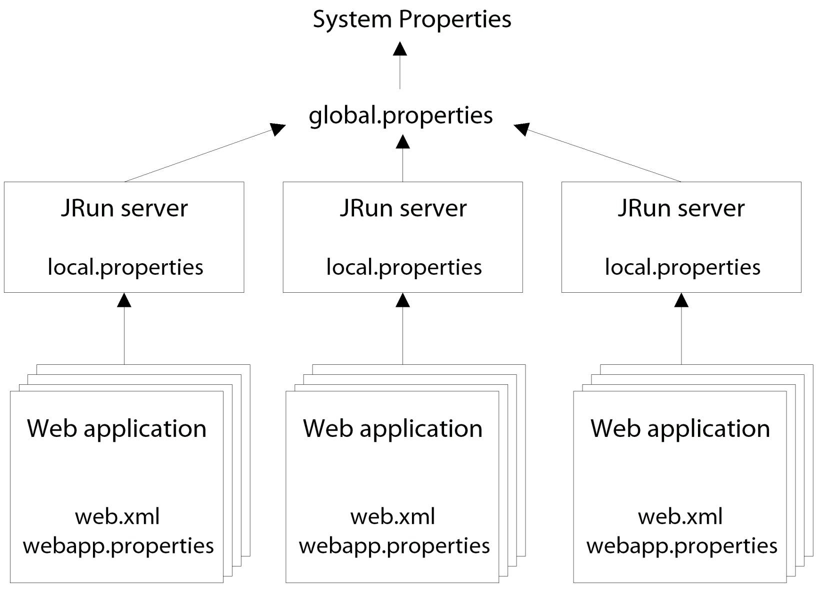
File Name	Location	Description
`descriptions.properties`	`\lib\`	Stores descriptions of many of the objects found in the property files.
`global.properties`	`\lib\`	Stores the highest level of values. The local.properties files default in values from global.properties.
`local.properties`	`\servers\`*`<server_name>`*`\`	Stores properties for each JRun server and all applications on that server. Overrides global properties.
`webapp.properties`	`\servers\`*`<server_name>`*`\`*`<app_name>`*`\WEB-INF\`	Optionally stores properties for a specific Web application. Overrides local and global properties. JRun only creates this file when you use the JMC to set application-specific settings.
`jvms.properties`	`\lib\`	Lists names and absolute paths to all JRun servers in this JRun installation.
`pass.properties`	`\lib\`	Contains encrypted password and permissions settings for all users.
`serial_number.properties`	`\lib\`	Stores JRun's serial number.
`users.properties`	`\lib\`	Contains a list of users and their passwords, as well as assignations of users to groups and groups to roles.
ejipt.properties	\lib\	Stores host information for the EJB engine.
deploy.properties	\servers\<*server_name*>\deploy	Stores server-level properties and is used by the Deploy tool to determine the beans to be deployed, host name, data sources, and connection limits.

Property Files		
File Name	**Location**	**Description**
<*bean_name*>.properties	*<Bean's interface and implementation directory>*	Stores security and bean description information. Similar information can be stored in the bean's XML descriptor file.
default.properties	*<Bean's interface and implementation directory>*	Stores the context information for this bean.

Understanding the Property File Hierarchy

JRun's property files are structured in a four-tier hierarchy: system, global, local, and application. Settings at the global level can be overridden at the local level, which in turn can be overridden at the application level. System settings cannot be overridden. If a setting is not overridden, the lower-level property file inherits the value from the higher-level property file.

The image below shows the four levels of JRun settings.



The system-level settings are computed at runtime. These settings cannot be overridden by editing the properties files or using the JMC. These values include:

```
jrun.server.rootdir
jrun.server.name
jrun.rootdir (UNIX only)
```

The `jrun.rootdir` system setting on Windows 95/98/NT/2000 is set by JRun's installation utility and is stored in the Windows registry.

The global properties consist of the `global.properties` file, plus general property files such as `pass.properties`, `jvms.properties`, and `users.properties`. Settings in these files apply to all JRun servers within a JRun installation. For example, each installation of JRun has one file containing passwords to access the JMC.

The `local.properties` files provide settings at the JRun server level of the JRun installation. These files inherit properties from the global and system levels but override them if they contain a local value.

The `webapp.properties` files provide settings at the Web application level. These settings override the local and global settings. JRun only creates these files when you use the JMC to set application-specific properties.

For example, assume the following assignments take place:

```
/default/local.properties     webapp=default_invoker
/admin/local.properties       webapp={default}
global.properties             webapp=invoker
```

Results:

The `webapp` variable for the `admin` server is set to `invoker`; for the `default` server, it is set to `default_invoker`.

Editing Property Files

There are some simple rules to follow when editing JRun property files. These rules are covered in the section below.

Syntax

Use the following syntax rules when making changes to JRun properties files.

- All parameters are set using `<parameter>=<value>`, with no spaces before or after the equals sign (=).
- Values are separated by either commas or semi-colons.
- Do not insert trailing or leading whitespace (spaces or tabs) on any line.
- Enclose variables in curly braces `{ }`.
- Use the pound sign (#) for commenting out lines.

Editing

Keep the following in mind when changing JRun properties files.

- Make a backup of the property file before editing it.
- Use a text editor and save property files as plain text with a `*.properties` extension.
- When editing a `local.properties` file, restart the associated JRun server.
- When editing the `global.properties` file, restart all JRun servers in that JRun installation.

Using variables

JRun makes heavy use of variables and placeholders, which are indicated by curly braces `{ }`, in the property files and in the JRun Management Console (JMC).

`{variable}` is replaced with a value at runtime. Variables and constants can be mixed in the same assignment:

```
jrun.services=scheduler,logging,monitor,{servlet.services},control
```

Variables and constants can also be mixed within the same value, as well:

```
logging.filename={jrun.rootdir}/logs/{jrun.server.name}-event.log
```

The most common variables you will encounter are `{jrun.rootdir}` and `{default}`, the latter really acting as a placeholder rather than a variable. These and other variables are discussed in the table below.

Using Variables

Variable	Description
`{jrun.rootdir}`	The `{jrun.rootdir}` variable is a system variable set during the installation. On UNIX systems, this value is computed when the JRun server process starts. On Windows 95/98/NT, the value is stored in the system registry once during installation. When launching the JVM executable associated with a JRun server, JRun includes the `-D` switch and passes `jrun.rootdir` with it's associated value. For example: `java -D jrun.rootdir=c:\Allaire\JRun\`
`{default-app.rootdir}`	All Web applications in JRun have a root directory, which can be dynamically addressed using this variable. This is the document root for serving application files.

Using Variables	
Variable	**Description**
`{default}`	A value of `{default}` indicates that the parameter is set in another property file. The most common scenario is that a parameter is set to a real value in `global.properties` and the parameter is set to `{default}` in `local.properties`. A blank assignment sets the variable to null. If you see `foo={default}` in `local.properties` and `foo=` in `global.properties`, JRun assigns a null value to `foo`. You should usually not encounter a situation where the same parameter is set to `{default}` or blank in both `local.properties` and `global.properties`.

Other Variables	Some dynamic system variables are available to you in the properties files. These include `{date}`, `{hour}`, `{day}`, `{month}`, or `{year}` and are used primarily in the log file output settings or file naming.
	Do not confuse objects with variables. For example, `logging.dispatchLogger.events` specifies the events dispatched by the dispatch logger as a comma-separated list. These events may look like system variables, but they are objects and you should not edit them in general. The default value is:
	`{logging.infoevent},{logging.debugevent},`
	`{logging.warningevent}, {logging.errorevent}`

Index